Proceedings

THIRTIETH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

September 30 - October 2, 1992

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

PROCEEDINGS

THIRTIETH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

Paul Van Dooren
Mark W. Spong
Conference Co-Chairs

Conference held
September 30 - October 2, 1992
Allerton House
Monticello, Illinois

Sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

Digitized by the Internet Archive
in 2026 with funding from
University of Illinois Urbana-Champaign

https://archive.org/details/proceedings1992alle_0

FOREWORD

The Thirtieth Annual Allerton Conference on Communication, Control, and Computing was held at Allerton House, Monticello, Illinois on September 30, October 1 and 2, 1992. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain papers presented at the Conference.

We are very grateful to Professor Jean-Claude Latombe of Stanford University for his plenary lecture "Assembly Planning." We are also grateful to Professor David Kuck of the University of Illinois for his Wednesday evening lecture, ‘‘Delivering Parallel Performance’’, and to Professor Thomas Huang, also of the University of Illinois, for his entertaining banquet address titled "Image Processing in the Popular Media." We would like to thank all the session chairs and the authors for their contributions. Special thanks goes to the organizers of invited sessions who took time from their busy schedules to put together sessions on subjects of great current interest.

We appreciate the support of Professor W. K. Jenkins, Director of the Coordinated Science Laboratory and Professor T. N. Trick, Head of the Department of Electrical and Computer Engineering. We would like to thank our colleagues, Professors B. Bamieh, M. T. Başar, T. U. Başar, Y. Bresler, D. Brown, B. E. Hajek, T. S. Huang, S. Hutchinson, D. L. Jones, S. Kang, P. R. Kumar, M. C. Loui, J. V. Medanic, S. Meyn, D. C. Munson, M. Orchard, W. R. Perkins, J. Ponce, M. B. Pursley, D. V. Sarwate, R. Sreenivas, P. Voulgaris, K. Zeger, for their generous assistance in reviewing papers and organizing the papers into sessions. We are also indebted to colleagues elsewhere who so willingly stepped in to assist with reviewing.

We are also grateful to Mr. Steven Williams and Ms. Susan Baxley for handling the business affairs of the Conference; to Mr. Robert Ebeling and Ms. Lilian Beck and their assistants for the printing of large numbers of programs and registration packets; to Mr. Jack Gladin and Mr. Bob McFarlane for their assistance in the photography and drafting; to Ms. Beverly Becker and the wonderful staff at Allerton House who so cheerfully accommodated our group; and to Dan Jordan who handles the shipping and sales of these *Proceedings* after the Conference. Most especially we want to thank Ms. Francie Bridges, Ms. Dixie Murphy, Ms. Janet Reese, and Ms. Lila Rhoades for their year-round assistance with the Conference in handling announcements, registration, program preparation, Allerton House arrangements, and the editing of these *Proceedings*. They are the real organizers of this Conference.

Paul Van Dooren

Mark Spong

TABLE OF CONTENTS

FOREWORD i

I-A: CODING

Chairman and Organizer: Daniel J. Costello, Jr.
University of Notre Dame

A FAMILY OF BCH CODES FOR THE LEE METRIC 1
R. M. Roth and P. H. Siegel

CODES FROM HYPERELLIPTIC CURVES 11
T. Yaghoobian and I. F. Blake

DEVELOPMENT OF TIGHTER ANALYTICAL BOUNDS ON THE ERROR PERFORMANCE OF CONVOLUTIONAL CODES 15
P. R. Molnar and S. G. Wilson

AN UPPER BOUND ON THE FREE DISTANCE OF DOUBLE MEMORY CONVOLUTIONAL CODES 20
D. G. Mills and D. J. Costello, Jr.

THE ERROR PROBABILITY OF EXPANSION CODES 27
A. S. Khayrallah

MAXIMUM LIKELIHOOD ERROR-TRAPPING DECODING OF BINARY CYCLIC CODES 37
W. Sung and J. T. Coffey

I-B: ADAPTIVE SYSTEMS

Chairman and Organizer: W. Kenneth Jenkins
University of Illinois at Urbana-Champaign

GENERAL STRUCTURAL SUBBAND DECOMPOSITION OF ADAPTIVE FILTERS FOR SYSTEM IDENTIFICATION PROBLEMS 39
S. Song, M. R. Petraglia, S. K. Mitra, and A. Mahalanobis

ADAPTIVE FILTER DESIGN FOR A HIGH PERFORMANCE ECHO CANCELLER 49
R. C. Younce and C. E. Rohrs

ON THE TRACKING CAPABILITIES OF OPTIMAL BOUNDING ELLIPSOID ALGORITHMS 50
Y. F. Huang and J. R. Deller, Jr.

AN ADAPTIVE INTERFERENCE CANCELLATION SYSTEM USING ACOUSTIC CHARGE TRANSPORT TECHNOLOGY 60
D. A. Fleisch, A. J. Vigil, A. W. Hull, and B. J. Hunsinger

AN EFFICIENT TWO-STAGE LATTICE IMPLEMENTATION OF A SECOND ORDER ADAPTIVE VOLTERRA FILTER 71
M. V. Dokic and P. M. Clarkson

I-C: LINEAR SYSTEMS

Chairman: Petros Voulgaris
University of Illinois at Urbana-Champaign

POSITIVE INVARIANCE OF SYMMETRIC POLYHEDRAL SETS FOR CONTINUOUS-TIME LINEAR SYSTEMS WITH APPLICATIONS 73
D. Liu and A. N. Michel

ON THE INVARIANCE OF CONTROLLABILITY INDICES OF LINEAR MIMO SYSTEMS 83
S. Bingulac and H. F. VanLandingham

ALTERNATING CONVEX PROJECTION METHOD FOR COVARIANCE CONTROL DESIGN 88
K. M. Grigoriadis and R. E. Skelton

VARIANCE AND BIAS CONFIDENCE CRITERIA FOR SYSTEM AND OBSERVER/KALMAN FILTER IDENTIFICATION 98
D.-H. Tseng, R. W. Longman, J.-N. Juang, and C.-W. Chen

THE DETERMINATION OF FEEDBACK GAINS FOR UNCERTIN LINEAR SYSTEMS USING HIGHER-ORDER SENSITIVITY MINIMIZATION 110
B. C. Fabien

INTERNAL MODELS IN TWO-DEGREES-OF-FREEDOM CONTROL SYSTEMS 112
O. R. Gonzalez

BLOCK DIAGONAL OUTPUT COVARIANCE CONSTRAINT PROBLEM FOR PERIODIC AND MULTIRATE SYSTEMS 115
G. Zhu and R. E. Skelton

I-D: CONCURRENT AND DISTRIBUTED COMPUTATION

Chairman: Michael C. Loui
University of Illinois at Urbana-Champaign

BIDIRECTIONAL DISTRIBUTED DEADLOCK DETECTION AND RESOLUTION ALGORITHM .. 117
Y. Shin and D. L. Epley

TIGHT LOWER BOUNDS ON FAILURE LOCALITY OF DISTRIBUTED SYNCHRONIZATION..... 127
M. Choy and A. K. Singh

TESTING AND DEBUGGING DISTRIBUTED PROGRAMS USING GLOBAL PREDICATES.......... 137
S. Venkatesan and B. Dathan

SHARED PROCESSORS VERSUS SHARED NETWORKS 147
D. P. Gruska

THE COMPLEXITY OF DECIDING BRANCHING BISIMILARITY FOR NORMED CONTEXT-FREE PROCESSES 157
D. Caucal, D. T. Huynh and L. Tian

II-A: COMMUNICATIONS

Chairman: Dilip V. Sarwate
University of Illinois at Urbana-Champaign

MPSK ERROR BOUNDS USING ALTERNATING FUNCTIONS 167
J. J. Komo and K. D. Barnett

A KALMAN FILTERING APPROACH TO INTERFERENCE SUPPRESSION IN DS/SS COMMUNICATION SYSTEMS 174
D. Krinsky and C. C. Lee

ON MAXIMIZING THE LINEAR SYSTEM OUTPUT ENERGY WITH A TIME- AND AMPLITUDE-LIMITED INPUT 184
J. S. Sadowsky

II-B: PARALLEL ALGORITHMS

Chairman: Michael M. Wu
Kuck and Associates, Inc.

EFFICIENT PARALLEL DYNAMIC PROGRAMMING 185
P. G. Bradford

PARALLEL ALGORITHMS FOR CONVEX BIPARTITE GRAPHS AND RELATED PROBLEMS..... 195
M. G. Andrews and D. T. Lee

PARALLEL EDGE COLORING APPROXIMATION 205
M. Fürer and B. Raghavachari

PARALLEL MULTISELECTION AND OPTIMAL MERGING 215
N. Deo, A. Jain and M. Medidi

GENERATING BINARY TREES IN PARALLEL 225
S. G. Akl and I. Stojmenovic

AN EFFICIENT PARALLEL PARTITIONING ALGORITHM 235
R. Sarnath

II-C: NUMERICAL METHODS FOR SIGNALS & SYSTEMS

Chairman and Organizer: Paul Van Dooren
Univ. of Illinois at Urbana-Champaign

NUMERICAL SOLUTION OF GENERALIZED LYAPUNOV EQUATION FOR DESCRIPTOR SYSTEMS 241
P. Misra

A QUADRATICALLY CONVERGENT LOCAL ALGORITHM ON MINIMIZING THE LARGEST EIGENVALUE OF A SYMMETRIC MATRIX 248
M. K. H. Fan

ITERATIVE IMAGE RESTORATION USING FFT-BASED PRECONDITIONERS 257
J. G. Nagy and R. J. Plemmons

A COMPUTATIONAL ALGORITHM FOR COVARIANCE CONTROL: CONTINUOUS-TIME CASE 267
T. Iwasaki, R. E. Skelton, and M. J. Corless

THE CONVERGENCE OF ASYNCHRONOUS ITERATIONS FOR THE FIXED POINT OF A SPLITTING OPERATOR 277
L. Li and T. Nakamura

II-D: ESTIMATION AND DETECTION

Chairman: Joseph A. O'Sullivan
Washington Univ.

MULTI-TARGET NARROWBAND DIRECTION FINDING AND TRACKING USING MOTION DYNAMICS.......... 279
A. Srivastava, N. Cutaia, M. I. Miller, J. A. O'Sullivan and D. L. Snyder

OPTIMUM QUANTIZATION IN SINGLE AND MULTISENSOR RANDOM SIGNAL DETECTION 289
R. S. Blum

SOME RESULTS ON DISTRIBUTED DETECTION SCHEMES USING MULTILEVEL QUANTIZATION.......... 299
A. Nasipuri and S. Tantaratana

CLASSIFICATION USING SET-VALUED KALMAN FILTERING AND RISK-SENSITIVE BAYES THEORY.......... 309
S. E. Budge and T. K. Moon

PARAMETER ESTIMATION FOR DISCONTINUITY-PRESERVING STOCHASTIC SIGNAL MODELS.......... 319
R. R. Schultz and R. L. Stevenson

AN EXTENSION OF GOOD'S ROUGHNESS PENALTY FOR NONPARAMETRIC DENSITY ESTIMATION.......... 329
D. L. Snyder, A. Lanterman and M. I. Miller

III-A: ROBUST CONTROL

Chairman: Bassam Bamieh
University of Illinois at Urbana-Champaign

ON THE ASYMPTOTIC STABILITY OF m-SWITCHED SYSTEMS.......... 330
P. Peleties and R. DeCarlo

ROBUST STABILITY OF SINGULARLY PERTURBED SYSTEMS.......... 340
Z. Shao and M. E. Sawan

EXPONENTIAL STABILIZATION OF UNCERTAIN SYSTEMS BY DYNAMIC FEEDBACK.......... 349
E. Yaz and A. Azemi

ROBUST STABILIZATION OF INTERVAL PLANTS BY STATE SPACE METHODS.......... 359
K. M. Grigoriadis, E. Yaz, and R. Skelton

IMPROVED SUFFICIENT CONDITIONS FOR THE STABILITY OF INTERVAL MATRICES.......... 369
K. Wang and A. N. Michel

ROBUST STABILITY AND PERFORMANCE ANALYSIS OF SYSTEMS WITH H_∞-NORM BOUNDED STRUCTURED UNCERTAINTIES.......... 371
Z. Shao and M. E. Sawan

III-B: GEOMETRIC ALGORITHMS AND VLSI

Chairman: Der-Tsai Lee
Northwestern University

CONSTRUCTION OF d-DIMENSIONAL HYPEROCTREES ON A HYPERCUBE MULTIPROCESSOR.......... 373
F. Dehne, A. Fabri, M. Nassar, A. Rau-Chaplin and R. Valiveti

OPTIMAL MESH ALGORITHMS FOR THE VORONOI DIAGRAM OF LINE SEGMENTS, VISIBILITY GRAPHS AND MOTION PLANNING IN THE PLANE 382
S. Rajasekaran and S. Ramaswami

IMPROVED RECURSIVE BISECTION LINE DRAWING ALGORITHMS 392
P. Graham, S. S. Iyengar, and S.-Q. Zheng

GUARDING POLYHEDRAL TERRAINS 402
P. Bose, T. Shermer, G. Toussaint, and B. Zhu

OPTIMAL ALGORITHMS FOR MINIMIZING TOTAL CHANNEL DENSITY IN CHANNEL-PLACEMENT PROBLEMS 404
M. Hossain, N. A. Sherwani, and S.-Q. Zheng

ALGORITHMS FOR CHANNEL ROUTING WITH INTERCHANGEABLE TERMINALS 414
S. Tragoudas

EMBEDDING COMPLETE BINARY TREES ON SQUARE MESHES WITH SMALL LINK CONGESTION 424
S.-K. Lee and H.-A. Choi

III-C: CELLULAR COMMUNICATIONS

Chairman: David Borth
Motorola, Inc.

CAPACITY IN A TWO CELL SPREAD SPECTRUM NETWORK 426
S. V. Hanly

DIVERSITY BASED DETECTORS IN MULTI-ACCESS NETWORKS 436
S. Kandala, E. S. Sousa and S. Pasupathy

A DISTRIBUTED ALGORITHM FOR POWER CONTROL IN CELLULAR RADIO SYSTEMS 446
S. A. Grandhi, R. Vijayan and D. J. Goodman

MAXIMIZING MARGINAL REVENUE IN GENERALIZED BLOCKING SERVICE NETWORKS.... 455
R. J. McEliece and K. N. Sivarajan

PERFORMANCE EVALUATION OF INDOOR MICROCELLULAR SYSTEMS 465
I. Argyropoulos and S. P. R. Kumar

III-D: SIGNAL PROCESSING

Chairman: Rick Blum
Lehigh University

APPLICATION OF NON-LINEAR TRANSFORMATIONS TO SIGNAL PROCESSING 475
J. E. Gray and B. S. McDole

ASYMPTOTIC STABILITY OF FIXED-POINT DIGITAL FILTERS USING GENERALIZED OVERFLOW ARITHMETIC 485
D. Liu and A. N. Michel

RECURSIVE DIRECTION FINDING VIA RLS 495
J. H. Clements III and J. T.-H. Lo

MAXIMUM LIKELIHOOD CLUSTERING WITH SIMULATED ANNEALING.................................... 505
K.-B. Eom and J. Park

RADAR TARGET RECOGNITION USING SHAPE MODELS 515
J. A. O'Sullivan, K. C. Du, R. S. Teichman, M. I. Miller, D. L. Snyder and V. C. Vannicola

WALD'S SPRT APPLIED TO SPEAKER VERIFICATION 524
M. A. Lund and C. C. Lee

IV-A: COMBINATORIAL ALGORITHMS

Chairman: David S. Atkinson
University of Illinois at Urbana-Champaign

THE POWER OF LOCAL OPTIMIZATION: APPROXIMATION ALGORITHMS FOR MAXIMUM-LEAF SPANNING TREE.. 533
H.-I. Lu and R. Ravi

PROVABLY GOOD ALGORITHMS FOR HEREDITARY PROPERTY PROBLEMS........................... 543
G. Chartrand, J. McCanna, N. A. Sherwani, M. Hossain and J. Hashmi

PROBABILISTIC ANALYSIS OF AN ENHANCED PARTITIONING ALGORITHM FOR THE STEINER TREE PROBLEM IN R^d 553
K. Kalpakis and A. T. Sherman

ON SOLVING PROBLEMS FOR p-CONTRACTABLE GRAPHS... 565
B. Ramamurthy and M. S. Krishnamoorthy

MINIMUM COVER AND SINGLE SOURCE SHORTEST PATH PROBLEMS FOR WEIGHTED INTERVAL GRAPHS AND CIRCULAR-ARC GRAPHS.. 575
O. H. Ibarra, H. Wang and Q. Zheng

PALINDROMIC TREES.. 585
B. De Iaco, F. Luccio and L. Pagli

FAST ALGORITHMS FOR INFERRING EVOLUTIONARY TREES.. 594
R. Agarwala, D. Fernandez-Baca, and G. Slutzki

IV-B: DISCRETE EVENT SYSTEMS

Chairman: Ramavarapu Sreenivas
University of Illinois at Urbana-Champaign

EXISTENCE AND DESIGN OF SUPERVISORS FOR VECTOR DISCRETE EVENT SYSTEMS........ 604
S. L. Chen and W. M. Wonham

HIERARCHICAL AND MODULAR CONTROL OF DISCRETE-EVENT SYSTEMS........................... 614
K. C. Wong and W. M. Wonham

MODULAR SUPERVISORY CONTROL OF TIMED DISCRETE-EVENT SYSTEMS 624
B. A. Brandin, W. M. Wonham, and B. Benhabib

ON THE SYNTHESIS OF SUPERVISORS FOR TIMED DISCRETE EVENT PROCESSES................. 633
S. D. O'Young

NECESSARY CONDITIONS FOR TRACKING TIMING PERTURBATIONS IN TIMED PETRI NETS 643
K. J. Williams, M. S. Andersland, J. A. Gannon, J. E. Lumpp, Jr. and T. L. Casavant

A FRAMEWORK FOR TIMED DISCRETE-EVENT SYSTEMS 650
T.-J. Ho and W. M. Wonham

IV-C: COMMUNICATION NETWORKS

Chairman: Bruce Hajek
University of Ilinois at Urbana-Champaign

ASSIGNMENT OF 2^k TRUNK GROUPS IN MULTICHANNEL SWITCHES USING GENERALIZED BINARY ADDRESS 652
H. Saidi, P. S. Min, and M. V. Hegde

DECENTRALIZED CONTROL OF NETWORKS: BAD NEWS SHOULD TRAVEL FAST, GOOD NEWS SLOWLY 662
R. K. Boel

PACKET ROUTING IN NETWORKS WITH LONG WIRES 664
R. I. Greenberg and H.-C. Oh

COMMUNICATIONS NETWORKS AS STOCHASTICALLY PERTURBED NONLINEAR SYSTEMS: A CAUTIONARY NOTE 674
R. S. Maier

MULTICAST IN ATM NETWORKS 682
M. Doar and N. G. Bean

IV-D: ROBOT CONTROL

Chairman: Victoria Coverstone-Carroll
University of Illinois at Urbana-Champaign

BILATERAL CONTROL OF TELEOPERATORS WITH TIME DELAY THROUGH A DIGITAL COMMUNICATION CHANNEL 692
G. M. H. Leung and B. A. Francis

HYBRID POSITION/FORCE CONTROL OF REDUNDANT ROBOT WITH CONSIDERATION OF MOTOR DYNAMICS 702
T. J. Tarn, A. K. Bejczy, and N. Xi

VISUALLY-GUIDED ROBOTIC MOTION TRACKING 712
M. Lei and B. K. Ghosh

STATE ESTIMATION FOR ELASTIC JOINT ROBOTS FROM MEASUREMENTS OF MOTOR ROTOR POSITIONS AND VELOCITIES 722
M. Jankovic

INDIRECT LEARNING CONTROL ALGORITHMS FOR TIME-VARYING AND TIME-INVARIANT SYSTEMS 731
R. W. Longman and Y. S. Ryu

ACTIVE CONTROL OF MACHINING PROCESSES USING IMAGE RECONSTRUCTION AND PROCESSING 742
D. DeVoe, L. Knox and G. Zhang

PLENARY ADDRESS GIVEN BY J.-C. LATOMBE

REASONING ABOUT MECHANICAL ASSEMBLY 751
R. H. Wilson and J.-C. Latombe

V-A: INFORMATION THEORY

Chairman: Ken Zeger
University of Illinois at Urbana-Champaign

INFORMATION-THEORETIC CAPACITY OF RANDOM MULTI-RECEIVER NETWORKS 782
S. V. Hanly and P. Whiting

A PHYSICALLY BASED APPROACH TO INFORMATION THEORY FOR THIN FILM MAGNETIC RECORDING 792
J. A. O'Sullivan, D. G. Porter, R. S. Indeck, and M. W. Muller

ON GENERATING RANDOM BITS FROM AN ARBITRARY RANDOM SOURCE 799
S. Vembu and S. Verdú

SOURCE CODING THEOREM AND ITS CONVERSE FOR DISCRETE TIME MS PERIODIC GAUSSIAN SOURCES 808
W.-J. Hwang and H. Derin

V-B: ROBOTICS & COMPUTER VISION

Chairmen and Organizers: Jean Ponce & Seth Hutchinson
University of Illinois at Urbana-Champaign

EXPERIMENTS IN HYPER-REDUNDANT MANIPULATION 818
G. S. Chirikjian and J. Burdick

KINEMATIC DESIGN OF FAULT TOLERANT MANIPULATORS 826
W. K. F. Au, C. J. J. Paredis and P. Khosla

COMPUTING A STATISTICAL DISTRIBUTION OF STABLE POSES FOR A POLYHEDRON 836
J. Wiegley, A. Rao, and K. Goldberg

MACHINE VISION SYSTEMS FOR SHAPE RECOVERY 844
S. K. Nayar

SENSOR PLANNING FOR REACTIVE ROBOTIC SYSTEMS 854
G. D. Hager and G. Grunwald

V-C: DATA STRUCTURES AND FOUNDATIONS

Chairman: Donna J. Brown
University of Illinois at Urbana-Champaign

MULTIDIMENSIONAL (a,b)-TREES: A CLOSE LOOK AT THE NEIGHBOR-SUPPORT PARADIGM 864
V. K. Vaishnavi and R. R. Mixon

A PROBABILISTIC ANALYSIS OF FIXED AND ELASTIC BUCKETS IN TRIES AND PATRICIA TREES 874
H. M. Mahmoud and T. Papadakis

INCORPORATING DEPENDENT AND MULTIMODE FAILURES INTO RELIABILITY EVALUATION OF EXTRA STAGE SHUFFLE-EXCHANGE MINs 884
J. L. Trahan, D. X. Wang and S. Rai

ON THE POWER OF DISCONTINUOUS APPROXIMATE COMPUTATIONS 893
K. Aberer and B. Codenotti

V-D: NEURAL NETWORKS & NONLINEAR DYNAMICS

Chairman: Andrew Noble
University of Illinois at Urbana-Champaign

OPTIMAL FILTERING BY RECURRENT NEURAL NETWORKS 903
J. T.-H. Lo

A HIERARCHICAL ALGORITHM FOR NEURAL TRAINING AND CONTROL 913
T.V. Theodosopoulos, M.S. Branicky, and M.M. Livstone

THE LOGISTIC LATTICE IN RANDOM NUMBER GENERATION 922
N. R. Wagner

A PARALLEL SCHEME FOR THE CELL-TO-CELL BASED ANALYSIS OF DYNAMIC SYSTEMS 932
L. Fortuna, M. L. Presti, G. Muscato and C. Vinci

INDEX OF AUTHORS 936

A Family of BCH Codes for the Lee Metric

RON M. ROTH
Computer Science Department, Technion, Haifa 32000, Israel

PAUL H. SIEGEL
IBM Research Division, Almaden Research Center, San Jose, CA 95120

Abstract — We show that each code in a certain class of BCH codes over $GF(p)$, specified by a code length $n \leq p^m - 1$ and a runlength $r \leq (p-1)/2$ of consecutive roots in $GF(p^m)$, has minimum Lee distance $\geq 2r$. For the very high-rate range these codes approach the sphere-packing bound on the minimum Lee distance. Furthermore, for a given r, the length range of these codes is twice as large as that attainable by Berlekamp's extended negacyclic codes. We present an efficient decoding procedure, based on Euclid's algorithm, for correcting up to $r-1$ errors and detecting r errors, that is, up to the number of Lee errors guaranteed by the designed minimum Lee distance $2r$. Bounds on the minimum Lee distance for $r \geq (p+1)/2$ are provided for the Reed-Solomon case i.e., when the BCH code roots are in $GF(p)$.

1 Introduction

The Lee metric [9],[16] was developed as an alternative to the Hamming metric for transmission of non-binary signals (usually taken from $GF(p)$) over certain noisy channels. The Lee distance $d_{\mathcal{L}}(x, y)$ between two elements x, y in $GF(p)$ is the smallest absolute value of any integer congruent, modulo p, to the difference $x - y$. Therefore, the Lee metric is "circular" when applied to $GF(p)$, and, for this reason, has on occasion been proposed for use in the context of phase modulation [1, Section 8.2]. Codes for the Lee metric were described first by Lee [9] and Ulrich [16], but perhaps the most important and well-known codes for the Lee metric are the negacyclic codes introduced by Berlekamp [1, Ch. 9], for which there is an efficient decoding procedure. The core of the decoding procedure is the application of the Berlekamp-Massey algorithm to a polynomial congruence similar to the key equation for BCH codes in the Hamming metric. Later, Chang and Wolf [2] devised a family of cyclic codes, for odd codeword lengths, with Lee-metric properties very similar to those of the negacyclic codes.

The Lee metric extends to symbols drawn from the alphabet of rational integers, where the Lee distance between symbols corresponds to the absolute value of their difference. Jinushi and Sakaniwa [7] recently reported a construction method for error-correcting codes over the integers that relies upon properties of generalized Hadamard matrices. Karabed and Siegel [8] observed that ensembles of integer sequences with higher-order nulls in the power spectral density at rational submultiples of the symbol frequency have substantial Lee-distance properties. The lower bound on the minimum Lee distance of such sequences generalizes a lower bound on the minimum Hamming distance for binary block codes with higher-order spectral density null at zero frequency, due to Immink and Beenker [6].

As mentioned in [8], the appearance of Newton's identities [11, p. 244] in the proof of the lower bounds on the minimum Lee distance for integer spectral-null codes suggested the existence of efficient, iterative decoding algorithms akin to those developed for BCH and Goppa codes in the Hamming metric. The details of such a decoding algorithm for spectral-null codes will be presented in this paper, but in the broader context of a BCH class of error-correcting codes for the Lee metric over $GF(p)$, as we now describe.

Motivated by the similarity in form of the moment equations characterizing integer block codes with a higher-order spectral null and the parity-check equations of BCH codes, we define in Section 2 a class of BCH codes over $GF(p)$, with each code specified by a code length $n \leq p^m - 1$ and a runlength r of consecutive roots in $GF(p^m)$.

In Section 3, we prove that, for those codes in this class satisfying the constraint that $r \leq (p-1)/2$, the minimum Lee distance is bounded from below by $2r$. The performance of these codes is compared with that of the negacyclic codes and their generalizations. For a given r and redundancy, the length range of the Lee-metric BCH codes is shown to be twice as large as that achieved by the negacyclic code construction. Furthermore, for small values of r and for sufficiently large m, Lee-metric BCH codes of length $n = p^m - 1$ approach the sphere-packing upper bound on the minimum Lee distance.

Section 4 addresses extensions and improvements of the $2r$ lower bound in the base-field case, which corresponds to Reed-Solomon codes of lengths $n \leq p - 1$ over $GF(p)$. In Section 4.1 we first extend the $2r$ lower bound to all values $r \leq n \leq p - 1$. Then, in Section 4.2, we provide a refined bound that, for the low-dimension (high-redundancy) case, becomes quadratic (rather than linear) in r. For $r \geq \frac{6}{7}p$, this bound improves upon the $2r$ lower bound.

Section 5 addresses the issue of decoding Lee-metric BCH codes. We develop a modified 'key equation' and present an efficient decoding procedure, based upon Euclid's algorithm, that can correct all error patterns up to Lee weight $r - 1$ and detects all error patterns of Lee weight r, for codes with designed minimum Lee distance $2r$. The decoding algorithm for the proposed codes seems to be simpler than Berlekamp's decoding algorithm for negacyclic codes [1, Algorithm 9.36].

Then, in Section 6, completing the circle, we return to the application that prompted this work, and use the decoding algorithm of Section 5 to develop an algebraic approach to the demodulation of integer-valued, spectral-null codes when used as matched-spectral-null codes on noisy partial-response channels where the Lee metric pertains.

2 Definitions

Let $C(n, r, \boldsymbol{\alpha}; p)$ be the (shortened) BCH code of length n over $GF(p)$ whose parity-check matrix is

$$H(n, r, \boldsymbol{\alpha}; p) \triangleq \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_1 & \alpha_2 & \ldots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \ldots & \alpha_n^2 \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{r-1} & \alpha_2^{r-1} & \ldots & \alpha_n^{r-1} \end{bmatrix},$$

where $\boldsymbol{\alpha} = [\,\alpha_1\, \alpha_2 \ldots \alpha_n\,]$ is the *locator vector*, consisting of distinct nonzero elements of the smallest field $GF(p^m)$ of size greater than n. Hence, a word $\mathbf{c} = [\,c_1\, c_2 \ldots c_n\,] \in GF(p)^n$ is in $C(n, r, \boldsymbol{\alpha}; p)$ if and only if it satisfies the following r parity-check equations over $GF(p^m)$:

$$\sum_{j=1}^{n} c_j \alpha_j^{\ell} = 0 \, , \quad \ell = 0, 1, \ldots, r-1 \, . \tag{1}$$

For $\ell \geq 1$, each parity-check equation in (1) translates into m equations over $GF(p)$. This gives the following well-known lower bound on the dimension k, or, rather, an upper bound on the redundancy $n - k$, of $C(n, r, \boldsymbol{\alpha}; p)$:

$$n - k \leq 1 + (r-1)m \, . \tag{2}$$

The codes $C(n,r,\boldsymbol{\alpha};p)$ for which $n = p^m - 1$ will be called *primitive.* In this case, $\boldsymbol{\alpha}$ is unique, up to permutation of coordinates, and, therefore, we shall sometimes use the short-hand notation $C(p^m-1,r;p)$ for $C(p^m-1,r,\boldsymbol{\alpha};p)$. For primitive codes, the bound (2) becomes

$$n-k \;\le\; 1 \;+\; (r-1)\log_p(n+1)\,. \tag{3}$$

A special interesting case is the base-field case $m = 1$ which corresponds to (generalized) Reed-Solomon codes of length $n \le p-1$ over $GF(p)$. In this case, the α_j are distinct nonzero elements of $GF(p)$, and the dimension of these codes is equal to $n-r$. The generator matrix of primitive base-field codes $C(p-1,r;p)$ has the form

$$G(p-1,r;p) \triangleq \left[\alpha_j^i\right]_{i=1,j=1}^{p-r-1,p-1}\,.$$

For an element $\alpha \in GF(p)$, we define the *Lee value* $|\alpha|$ by

$$|\alpha| \triangleq \begin{cases} \alpha & \text{when } \alpha = 0,1,\ldots,\frac{p-1}{2} \\ p-\alpha & \text{when } \alpha = \frac{p+1}{2},\frac{p+3}{2},\ldots,p-1 \end{cases}.$$

The elements $0,1,\ldots,(p-1)/2$ of $GF(p)$ will be referred to as the 'positive' elements of the field. The rest of the elements are the 'negative' ones.

For a vector $\mathbf{c} = [\,c_1\,c_2\,\ldots\,c_n\,]$ over $GF(p)$, we define the *Lee weight* by $\|\mathbf{c}\| \triangleq \sum_{j=1}^n |c_j|$ (summation taken over the integers). The *Lee distance* between two vectors in $GF(p)^n$ is defined as the Lee weight of their difference. The *minimum Lee distance* of a subset X of $GF(p)^n$ is the minimum Lee distance between any pair of distinct vectors in X. Since $C(n,r,\boldsymbol{\alpha};p)$ is an additive subgroup of $GF(p)^n$, the minimum Lee distance of $C(n,r,\boldsymbol{\alpha};p)$, denoted $d_{\mathcal{L}}(n,r,\boldsymbol{\alpha};p)$, is also the minimum Lee weight of any nonzero codeword in $C(n,r,\boldsymbol{\alpha};p)$.

Given a 'transmitted' word $\mathbf{c} \in GF(p)^n$ (say, a codeword in $C(n,r,\boldsymbol{\alpha};p)$) and a 'received' word $\mathbf{y} \in GF(p)^n$, the error vector is defined by $\mathbf{e} \triangleq \mathbf{y} - \mathbf{c}$. The number of *Lee errors* is given by $\|\mathbf{e}\|$; that is, the number of Lee errors is the smallest number of additions of ± 1 to the coordinates of the transmitted codeword $\mathbf{c}$ which yield the received word $\mathbf{y}$. Since the Lee weight satisfies the triangle inequality, using a code of minimum Lee distance $d_{\mathcal{L}}$ allows to correct any pattern of up to $(d_{\mathcal{L}}-1)/2$ Lee errors.

One of the applications that motivated this work was analyzing the correction capability of matched-spectral-null trellis codes for partial-response channels [8]. These codes can be modeled as sets of vectors $\mathbf{c} = [\,c_1\,c_2\,\ldots\,c_n\,]$ over the integer ring $\mathcal{Z}$ that satisfy the set of constraints

$$\sum_{j=1}^{n} (j-1)^\ell\, c_j \;=\; 0\,, \quad \ell = 0,1,\ldots,r-1 \tag{4}$$

(where $0^0 \triangleq 1$), for some prescribed order r of the spectral null at zero frequency. The constraints in (4) are equivalent, in turn, to

$$\sum_{j=1}^{n} j^\ell\, c_j \;=\; 0\,, \quad \ell = 0,1,\ldots,r-1\,.$$

Hence, along with the codes $C(n,r,\boldsymbol{\alpha};p)$, we shall be interested also in additive subgroups $C(n,r,\boldsymbol{\alpha})$ of $\mathcal{Z}^n$ consisting of words $\mathbf{c} \in \mathcal{Z}^n$ that satisfy the constraint $H(n,r,\boldsymbol{\alpha})\,\mathbf{c} \;=\; \mathbf{0}$, where $\boldsymbol{\alpha} = [\,\alpha_1\,\alpha_2\,\ldots\,\alpha_n\,]$ is a locator vector of distinct integers entries $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_n$ and $H(n,r,\boldsymbol{\alpha}) \triangleq \left[\alpha_j^\ell\right]_{\ell=0,j=1}^{r-1,n}$. When $\boldsymbol{\alpha} = [\,1\,2\,\ldots\,n\,]$ we shall use the shorter notation $C(n,r)$ for $C(n,r,\boldsymbol{\alpha})$.

Defining the Lee value of an integer as its (conventional) absolute value, the definition of the Lee weight of an integer vector, as well as the minimum Lee distance of any subset of $\mathcal{Z}^n$, is extended in a natural way. The minimum Lee distance of $C(n, r, \boldsymbol{\alpha})$ will be denoted by $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha})$.

3 The $2r$ lower bound

In [8], a lower bound $d_{\mathcal{L}}(n, r) \geq 2r$ on the minimum Lee distance of $C(n, r)$ was derived. The proof was a slight generalization of an argument, based upon Newton's identities, that was used in [6] to bound from below the minimum Hamming distance of binary codes with rth order spectral null at zero frequency. In fact, the very same proof can be used to show the more general lower bound $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}) \geq 2r$. The following theorem states that the $2r$ lower bound, with certain necessary restrictions on r, applies also to $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$.

Theorem 1.

$$d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p) \geq \begin{cases} 2r & \text{for } r \leq (p-1)/2 \\ p & \text{for } r \geq (p+1)/2 \end{cases} .$$

This bound is, in a way, the analog of the BCH lower bound $r + 1$ on the minimum Hamming distance of $C(n, r, \boldsymbol{\alpha}; p)$, although the proof of the $2r$ lower bound is slightly more complicated.

At this point, it is worthwhile comparing the performance of $C(n, r, \boldsymbol{\alpha}; p)$ in the Lee space with that of negacyclic codes [1, Ch. 9]. The latter, or, rather, generalized versions thereof, are defined as codes of length $n \leq (p^m - 1)/2$ over $GF(p)$ whose parity-check matrix is of the form

$$H_{\text{negacyclic}}(n, r, \boldsymbol{\alpha}; p) \triangleq \left[\alpha_j^{2\ell-1}\right]_{\ell=1, j=1}^{r-1, n} , \tag{5}$$

where $\boldsymbol{\alpha} = [\,\alpha_1\, \alpha_2\, \ldots\, \alpha_n\,]$ consists of distinct nonzero elements $\alpha_j \in GF(p^m)$ such that $\alpha_j + \alpha_\ell \neq 0$ for all j and ℓ. For $r \leq (p-1)/2$, the known lower bound on the minimum Lee distance of negacyclic codes is $2r - 1$ [1, Ch. 9], and this bound becomes $2r$ if we extend the codes by adding an all-one row to their parity-check matrix. The upper bound on the redundancy of these extended codes is equal to the corresponding bound (2) for $C(n, r, \boldsymbol{\alpha}; p)$. However, given m (dictated by specifications on r and redundancy constraints), the maximum attainable length of extended negacyclic codes is only half the maximum length of $C(n, r, \boldsymbol{\alpha}; p)$. As we shall see in Section 5, the decoding algorithm of $C(n, r, \boldsymbol{\alpha}; p)$ appears to be simpler than Berlekamp's decoding algorithm for the negacyclic case.

We end this section by exhibiting the near-optimality of the primitive codes $C(p^m - 1, r; p)$ for sufficiently small values of r.

The following result follows from the Lee-metric sphere-packing bound of Golomb and Welch [4].

Theorem 2. *A code over $GF(p)$ of length n, size p^k, and minimum Lee distance $\geq 2r - 1$ for some $r \leq (p+1)/2$ must satisfy the inequality*

$$(r-1)\left(\log_p(n - r + 2) - \log_p(r-1)\right) \leq n - k .$$

Return now to the code $C(p^m - 1, r; p)$ where, for $r \leq (p-1)/2$, we have $d_{\mathcal{L}}(p^m - 1, r; p) \geq 2r - 1$, thus conforming to the definition of r in Theorem 2. It is easy to verify that the lower bound of Theorem 2 on the redundancy $n-k$ approaches the upper bound on $n-k$ given in (3) when $\log_p r$ is much smaller than $m = \log_p(n+1)$. This would be the case when, for instance, we keep p, and therefore the range of r, fixed and let $n = p^m - 1$ go to infinity.

4 Lower bounds for the base-field case

Among the finite-field codes $C(n, r, \boldsymbol{\alpha}; p)$, the base-field codes are of some special interest in that they allow us to obtain bounds on the integer codes $C(n, r, \boldsymbol{\alpha})$ as well. In particular, for any code $C(n, r, \boldsymbol{\alpha})$ with $\boldsymbol{\alpha} = [\alpha_1\, \alpha_2 \ldots \alpha_n]$, $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_n$, and for every prime $p > \alpha_n \geq n$, we have $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}) \geq d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$. This is due to the fact that any nonzero codeword $\mathbf{c} \in C(n, r, \boldsymbol{\alpha})$ of minimum Lee weight must have at least one entry which is not divisible by p, and reducing such a codeword modulo p results in a nonzero codeword of $C(n, r, \boldsymbol{\alpha}; p)$ whose Lee weight is at most $\|\mathbf{c}\|$. Hence, any lower bound on $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$ implies one for $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha})$. The converse, of course, is not necessarily true.

4.1 Extending the $2r$ lower bound for the base-field case

As we pointed out in Section 3, the $2r$ lower bound does not hold in general for all values of r for any code $C(n, r, \boldsymbol{\alpha}; p)$; however, it does hold for all r in the base-field case $n \leq p-1$. We remark that for fairly large values of r, say, $r \geq (p+1)/2$, we believe that the true value of $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$ is much greater than $2r$. Our conjecture is based on the lower bounds on $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$ given in the next section, where we state that in the high-redundancy range, the lower bound on $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p)$ becomes quadratic, rather than linear, in r. Still, $2r$ is the best lower bound we have for values of r up to around $\frac{6}{7}p$. Furthermore, although we present a substantial improvement on the $2r$ lower bound for the high-redundancy range, we have yet to find an efficient way to decode the number of correctable errors guaranteed by that bound.

The $2r$ lower bound for the base-field case takes the following form.

Theorem 3. *For* $r \leq n \leq p-1$,

$$d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p) \;\geq\; 2r\;.$$

4.2 The low-dimension case

We turn now to improve the $2r$ lower bound for base-field codes in the low-dimension range. Since each base-field code $C(n, r, \boldsymbol{\alpha}; p)$ is a shortened code of $C(p-1, r; p)$, it suffices to consider only the primitive base-field case, bearing in mind that for $n \leq p-1$, $d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p) \geq d_{\mathcal{L}}(p-1, r; p)$.

Theorem 4. *For* $n \leq p-1$,

$$d_{\mathcal{L}}(n, r, \boldsymbol{\alpha}; p) \;\geq\; \frac{r+1}{2} \;+\; \frac{(r+1)^2}{4(p-1-r)}\;.$$

It is easy to check that the bound of Theorem 4 supersedes the $2r$ lower bound for $r \geq \frac{6}{7}p$. Furthermore, when $r = p - O(1)$, the lower bound of Theorem 4 becomes quadratic in r.

The following theorem, due to Mazur [13], improves on Theorem 4 for the *very* low dimension case.

Theorem 5. (Mazur [13]).

$$d_{\mathcal{L}}(p-1, p-1-k; p) \geq \frac{p^2-1}{4} - \frac{1}{4}(k-1) \cdot p^{3/2} .$$

In particular, Theorem 5 yields a quadratic lower bound for $r = p - O(\sqrt{p})$.

5 Decoding algorithm

In this section, we present a decoding procedure for $C(n, r, \boldsymbol{\alpha}; p)$, based upon Euclid's algorithm, that will correct all errors up to Lee weight $r-1$ and detect all errors of Lee weight r whenever the $2r$ lower bound applies (that is, when $r \leq (p-1)/2$ or when $r \leq n \leq p-1$). It is straightforward to adapt this algorithm to the integer codes $C(n, r, \boldsymbol{\alpha})$.

We first establish some notation. Let $\mathbf{c} = [\, c_1\, c_2 \, \ldots \, c_n \,]$ denote the 'transmitted' codeword and $\mathbf{y} = [\, y_1 \, y_2 \, \ldots \, y_n \,]$ denote the 'received' word, with the error vector given by $\mathbf{e} = [\, e_1 \, e_2 \, \ldots \, e_n \,] \triangleq \mathbf{y} - \mathbf{c}$. The corresponding 'positive' error vector $\mathbf{e}^+ = [\, e_1^+ \, e_2^+ \, \ldots \, e_n^+ \,]$, is defined by setting $e_j^+ = e_j$ if $e_j \in \{0, 1, \ldots, (p-1)/2\}$ and $e_j^+ = 0$ otherwise. Similarly, we define the 'negative' error vector $\mathbf{e}^- = [\, e_1^- \, e_2^- \, \ldots \, e_n^- \,]$, with $e_j^- = |e_j|$ if $e_j \in \{(p+1)/2, (p+3)/2, \ldots, p-1\}$ and $e_j^- = 0$ otherwise. The error vector can then be decomposed as $\mathbf{e} = \mathbf{e}^+ - \mathbf{e}^-$.

Given a locator vector $\boldsymbol{\alpha} = [\, \alpha_1 \, \alpha_2 \, \ldots \, \alpha_n \,]$, over $GF(p^m)$, we define the syndrome values S_ℓ of an error vector $\mathbf{e} = [\, e_1 \, e_2 \, \ldots \, e_n \,]$ in the standard way,

$$S_\ell = \sum_{j=1}^{n} e_j \alpha_j^\ell , \qquad 0 \leq \ell < \infty .$$

The formal syndrome series $S(x)$ is then defined as

$$S(x) = \sum_{\ell=1}^{\infty} S_\ell x^\ell .$$

(Note that the constant term corresponding to S_0 is excluded from $S(x)$.)

When the transmitted word $\mathbf{c}$ belongs to $C(n, r, \boldsymbol{\alpha}; p)$, the first r syndrome values S_ℓ can be determined from the received vector $\mathbf{y}$. Specifically,

$$S_\ell = \sum_{j=1}^{n} e_j \alpha_j^\ell = \sum_{j=1}^{n} y_j \alpha_j^\ell , \qquad 0 \leq \ell < r .$$

Therefore, when $\mathbf{c} \in C(n, r, \boldsymbol{\alpha}; p)$, the formal syndrome series $S(x)$ is in effect known modulo x^r.

It will be convenient to define the positive syndrome values S_ℓ^+ and the negative syndrome values S_ℓ^- of the error vector $\mathbf{e}$ by

$$S_\ell^+ = \sum_{j=1}^{n} e_j^+ \alpha_j^\ell \quad \text{and} \quad S_\ell^- = \sum_{j=1}^{n} e_j^- \alpha_j^\ell , \qquad 0 \leq \ell < \infty ,$$

with the associated formal syndrome series

$$S^+(x) = \sum_{\ell=1}^{\infty} S_\ell^+ x^\ell \quad \text{and} \quad S^-(x) = \sum_{\ell=1}^{\infty} S_\ell^- x^\ell .$$

Similarly, we define the positive and negative error-locator polynomials $\sigma^+(x)$ and $\sigma^-(x)$ by

$$\sigma^+(x) = \prod_{j=1}^{n} (1 - \alpha_j x)^{e_j^+} \quad \text{and} \quad \sigma^-(x) = \prod_{j=1}^{n} (1 - \alpha_j x)^{e_j^-} .$$

with $\sigma^+(x)$ and $\sigma^-(x)$, respectively.

Finally, we introduce the error-locator ratio,

$$\rho(x) = 1 + \sum_{i=1}^{\infty} \rho_i x^i = \frac{\sigma^+(x)}{\sigma^-(x)} .$$

Recalling that the formal syndrome series $S(x)$ is equal to $S^+(x) - S^-(x)$, we can apply Newton's identities [11, p. 244] to $S^+(x)$ and $S^-(x)$ to obtain the following relation between the error-locator ratio $\rho(x)$ and the formal syndrome series $S(x)$:

$$\rho(x)\, S(x) + x\, \rho'(x) = 0 . \tag{6}$$

Let $\phi(x)$ be the polynomial over $GF(p^m)$ defined by $\phi(x) = 1 + \sum_{i=1}^{r-1} \rho_i x^i$; that is, $\phi(x)$ is the unique polynomial of degree less than r satisfying

$$\phi(x) \equiv \rho(x) \pmod{x^r} .$$

From (6) we obtain

$$\phi(x)\, S(x) + x\, \phi'(x) \equiv 0 \pmod{x^r} , \tag{7}$$

which, in turn, can be rewritten explicitly as

$$S_i + \sum_{\ell=1}^{i-1} \rho_\ell S_{i-\ell} + i\, \rho_i = 0 , \qquad 1 \le i < r . \tag{8}$$

Knowing the syndrome values $S_1, S_2, \ldots, S_{r-1}$ from the received word $\mathbf{y}$, and noting that, for $r \le p$, the index i in (8) ranges over invertible integers modulo p, we can apply Equation (8) iteratively to solve (uniquely) for the values ρ_i for $i = 1, 2, \ldots, r-1$. Furthermore, the mapping $[\, S_1\, S_2\, \ldots\, S_{r-1} \,] \mapsto [\, \rho_1\, \rho_2\, \ldots\, \rho_{r-1} \,]$, induced by (8), is one-to-one. Hence, when the $2r$ lower bound applies, distinct error vectors $\mathbf{e}$ of Lee weight smaller than r correspond to distinct syndrome vectors $[\, S_0\, S_1\, S_2\, \ldots\, S_{r-1} \,]$ and, therefore, to distinct pairs $(S_0, \phi(x))$.

The following theorem summarizes a few properties of the error-locator polynomials $\sigma^+(x)$ and $\sigma^-(x)$.

Theorem 6. *Given a code $C(n, r, \boldsymbol{\alpha}; p)$ and an error vector $\mathbf{e}$ of Lee weight smaller than r, let $\sigma^+(x)$ and $\sigma^-(x)$ be the positive and negative error-locator polynomials, respectively, associated with $\mathbf{e}$, and let $\phi(x)$ be the polynomial defined by (7) for the syndrome vector $[\, S_0\, S_1\, \ldots\, S_{r-1} \,]^T = H(n, r, \boldsymbol{\alpha}; p)\, \mathbf{e}$. Then,*

(i) $\sigma^-(x)\, \phi(x) \equiv \sigma^+(x) \pmod{x^r}$;
(ii) $\deg \sigma^+ + \deg \sigma^- < r$;
(iii) $\gcd(\sigma^+, \sigma^-) = 1$;
and —
(iv) $\deg \sigma^+ - \deg \sigma^- \equiv S_0 \pmod{p}$.

Properties (i)–(iv) will serve as the 'key equations' for our decoding algorithm. We now aim at stating a result which is somewhat of a converse to Theorem 6 and which will allow us to use these key equations to compute the error-locator polynomials $\sigma^+(x)$ and $\sigma^-(x)$ in an efficient way by application of Euclid's algorithm. Euclid's algorithm has also been used to decode BCH codes and Goppa codes in the Hamming metric, as described in [11, Ch. 12],[14, Ch. 8].

The following converse to Theorem 6 provides the foundation for the decoding algorithm for the case $r \leq (p-1)/2$.

Theorem 7. *Given a code $C(n, r, \boldsymbol{\alpha}; p)$ with $r \leq (p-1)/2$ and with $\boldsymbol{\alpha}$ over $GF(p^m)$, let $\mathbf{e}$ be an error vector of Lee weight smaller than r and let S_0 and $\phi(x)$ be as in Theorem 6.*

(a) There is a unique (up to scalar normalization) pair of polynomials $\sigma^+(x)$ and $\sigma^-(x)$ over $GF(p^m)$ which satisfy properties (i)–(iv) of Theorem 6.

(b) With the proper scaling, the polynomials in (a) are the positive and negative error-locator polynomials, respectively, associated with $\mathbf{e}$.

(c) The polynomials $\sigma^+(x)$ and $\sigma^-(x)$ are given by

$$\sigma^+(x) = c \cdot R_s(x) \quad \text{and} \quad \sigma^-(x) = c \cdot T_s(x) \,,$$

where $R_i(x)$ and $T_i(x)$ are obtained, as in [14, Section 8.4], from the application of Euclid's algorithm to the polynomials $A(x) = x^r$ and $B(x) = \phi(x)$, and s is the unique index i for which

$$\deg R_s - \deg T_s = \begin{cases} S_0 & \text{if } S_0 = 0, 1, \ldots, r-1 \\ S_0 - p & \text{if } S_0 = p-r+1, p-r+2, \ldots, p-1 \end{cases} . \tag{9}$$

When $r \leq (p-1)/2$, there is a nonempty range of values of S_0 which corresponds to detectable but uncorrectable error patterns. Uncorrectable errors are detected also when the unique polynomials $\sigma^+(x)$ and $\sigma^-(x)$, if any, obtained by Theorem 7(c), violate the degree property (ii) of Theorem 6, or when these polynomials do not factor into linear terms $1 - \alpha_j x$ for elements α_j in the locator vector $\boldsymbol{\alpha}$. Uncorrectable errors will always be detected when the Lee weight of the error vector is exactly r.

Having determined the error-locator polynomials $\sigma^+(x)$ and $\sigma^-(x)$ by Theorem 7(c), we can now solve for the error vector $\mathbf{e} = [\, e_1\, e_2\, \ldots\, e_n \,]$ using the following modified Chien search (compare with [1, Algorithm 9.36]). For $j = 1, 2, \ldots, n$ we set $e_j = a$ (respectively, $e_j = -a$), where a is the smallest integer $i \geq 0$ for which the ith order formal derivative

$$\left(\sigma^+(x)\right)^{(i)} = \sum_{\ell \geq i} \ell(\ell-1)\cdots(\ell-i+1)\, \sigma_\ell^+ x^{\ell-i}$$

of $\sigma^+(x)$ (respectively, of $\sigma^-(x)$) does not vanish at $x = \alpha_j^{-1}$. (Since we expect to have multiplicities not greater than $(p-1)/2$ in the correct error-locator polynomials, the above test, using formal derivatives, does indeed provide the correct multiplicity. See [10, pp. 303–305].)

For the base-field case, in the range $r \geq (p+1)/2$, a complication arises from the fact that the stopping rule (9) might become ambiguous. In fact, part (a) of Theorem 7 no longer holds. However, a modification of Theorem 7 shows that the ambiguity may always be resolved into a unique solution.

The following is an outline of the decoding algorithm for $C(n, r, \boldsymbol{\alpha}; p)$ with $\boldsymbol{\alpha} = [\, \alpha_1\, \alpha_2\, \ldots\, \alpha_n \,]$. The input to the algorithm is the received word $[\, y_1\, y_2\, \ldots\, y_n \,]$, and the algorithm produces the error vector $[\, e_1\, e_2\, \ldots\, e_n \,]$, or returns an 'uncorrectable error' flag.

1. Compute the syndrome values $S_\ell \leftarrow \sum_{j=1}^{n} y_j \alpha_j^\ell$, $0 \le \ell < r$.
2. Compute the polynomial $\phi(x) = 1 + \sum_{i=1}^{r-1} \rho_i x^i$ using the recurrence
$$\rho_i \leftarrow -\frac{1}{i}\left(S_i + \sum_{\ell=1}^{i-1} \rho_\ell S_{i-\ell}\right), \qquad 1 \le i < r .$$
3. Apply Euclid's algorithm to the polynomials $A(x) = x^r$ and $B(x) = \phi(x)$ to obtain pairs of polynomials (R_i, T_i), $i = 0, 1, 2, \ldots$, until $\deg R_i - \deg T_i \le S_0 - p$.
4. For integers s for which $\deg R_s - \deg T_s \in \{S_0,\ S_0 - p\}$ and $\deg R_s + \deg T_s < r$ do:
    (a) let $\sigma^+(x) \leftarrow R_s(x)$ and $\sigma^-(x) \leftarrow T_s(x)$;
    (b) using formal derivatives find, for $j = 1, 2, \ldots, n$, the multiplicity e_j^+ of α_j^{-1} in $\sigma^+(x)$ and the multiplicity e_j^- of α_j^{-1} in $\sigma^-(x)$;
    (c) if $\sum_{j=1}^{n} e_j^+ = \deg \sigma^+$ and $\sum_{j=1}^{n} e_j^- = \deg \sigma^-$, set $e_j \leftarrow e_j^+ - e_j^-$.
5. If no such integers s exist, or if the values e_j were not set in step 4c, return an 'uncorrectable error' flag.

The decoding method we have just described for codes over $GF(p)$ is easily adapted for the integer codes $C(n, r, \boldsymbol{\alpha})$. In this case all operations will be carried out in the rational field, and we will have the stopping rule $\deg R_i - \deg T_i = S_0$ in Theorem 7(c).

6 Algebraic decoding of integer spectral-null codes

As was mentioned in Section 4, the $2r$ lower bound on the minimum Lee distance for the base-field codes $C(n, r, \boldsymbol{\alpha}; p)$ implies such a bound for the codes $C(n, r, \boldsymbol{\alpha})$ over the integer ring. In particular, the bound applies to codes with an rth-order spectral null at zero frequency [6],[8],[3] (see Equation (4)).

One application of integer spectral-null codes is to improving the reliability of information transmission over noisy partial-response channels. As shown in [8], the application of a code with Kth-order spectral null at zero frequency to a partial-response channel with Lth-order spectral null at zero frequency (i.e., transfer polynomial $h(D)$ divisible by $(1-D)^L$) ensures a minimum Lee distance no smaller than $2(K+L)$. When used in this context, the code is referred to in [8] as a matched-spectral-null code. We will consider the integer codes $C(n, K)$ for transmission over the channel $h(D) = (1-D)^L$. During the transmission process, a codeword of $C(n, K)$ is sent through the channel, followed by L consecutive zeros. Assuming that the initial channel memory is all-zero, the corresponding noiseless output words in the channel will be codewords of $C(n+L, K+L)$ i.e., they will have a $(K+L)$th-order spectral null at zero frequency.

When the channel noise samples are independent, and identically distributed according to a bilateral exponential density,
$$f(x) = \frac{\gamma}{2} e^{-\gamma|x|} ,$$
having zero mean and variance equal to $2\gamma^{-2}$, maximum-likelihood decoding is equivalent to finding a channel noiseless output word which is at the smallest Lee distance from the received word. Since the noiseless output words are codewords of $C(n+L, K+L)$, the algorithm of Section 5, when applied to $C(n+L, K+L)$, performs an efficient decoding with respect to the Lee metric for all error patterns in such a channel with Lee weight up to $K+L-1$.

Acknowledgment

The authors thank Noga Alon for helpful discussions. We also wish to thank Jack Wolf, Dave Forney, Dennis Howe, and Solomon Golomb for useful comments.

References

[1] E.R. BERLEKAMP, *Algebraic Coding Theory,* Revised Edition, Aegean Park Press, Laguna Hills, California, 1984.

[2] J.C.-Y. CHIANG, J.K. WOLF, *On channels and codes for the Lee metric, Inform. Control,* Vol. 19, No. 2 (September 1971), pp. 159–173.

[3] E. ELEFTHERIOU, R. CIDECIYAN, *On codes satisfying Mth order running digital sum constraints, IEEE Trans. Inform. Theory,* Vol. IT-37, No. 5 (September 1991), pp. 1294–1313.

[4] S.W. GOLOMB, L.R. WELCH, *Perfect codes in the Lee metric and the packing of polyominoes, SIAM J. Appl. Math.,* Vol. 18, No. 2 (January 1970), pp. 302–317.

[5] K.A.S. IMMINK, *Coding Techniques for Digital Recorders,* Prentice-Hall, London, 1991.

[6] K.A.S. IMMINK, G. BEENKER, *Binary transmission codes with higher order spectral zeros at zero frequency, IEEE Trans. Inform. Theory,* Vol. IT-33, No. 3 (May 1987), pp. 452–454.

[7] H. JINUSHI, K. SAKANIWA, *A construction method for multilevel error-correcting codes based on absolute summation weight, Abstracts of 1990 IEEE Int. Symp. Info. Th.,* San Diego, CA (January 1990), p. 87.

[8] R. KARABED, P.H. SIEGEL, *Matched spectral-null codes for partial-response channels, IEEE Trans. Inform. Theory,* Vol. 37, No. 3, Part II (May 1991), pp. 818–855.

[9] C.Y. LEE, *Some properties of nonbinary error-correcting codes, IRE Trans. Inform. Theory,* Vol. IT-4, No. 4 (June 1958), pp. 77–82.

[10] R. LIDL, H. NIEDERREITER, *Finite Fields,* Addison-Wesley, Reading, Massachusetts, 1983.

[11] F.J. MACWILLIAMS, N.J.A. SLOANE, *The Theory of Error-Correcting Codes,* North-Holland, Amsterdam, 1977.

[12] J.L. MASSEY, *Shift register synthesis and BCH decoding, IEEE Trans. Inform. Theory,* Vol. IT-15, No. 1 (January 1969), pp. 122–127.

[13] L.E. MAZUR, *Codes correcting errors of large weight in Lee metric, Problems Inform. Trans.,* Vol. 9, No. 4 (1973), pp. 277–281 (translated from Russian).

[14] R.J. MCELIECE, *The Theory of Information and Coding,* Addison-Wesley, Reading, Massachusetts, 1977.

[15] C. SATYANARAYANA, *Lee metric codes over integer residue rings, IEEE Trans. Inform. Theory,* Vol. IT-25, No. 2 (March 1979), pp. 250–254.

[16] W. ULRICH, *Non-binary error correction codes, Bell Sys. Tech. J.,* Vol. 36, No. 6 (November 1957), pp. 1341–1387.

CODES FROM HYPERELLIPTIC CURVES

by

Tomik Yaghoobian and Ian F. Blake
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario

ABSTRACT

Codes derived from curves in algebraic geometries over finite fields have received considerable attention over the past decade. The crucial parameters of length, dimension and minimum distance are determined by the Hasse-Weil and Riemann-Roch theorems. These are deep theorems of algebraic geometry and it is interesting to consider the possibility of determining similar codes without resort to such machinery. Such an approach is considered here for the class of codes derived from hyperelliptic curves. It is shown by elementary means that codes with parameters $n \leq q+1+2g\sqrt{q}$, dimension k and minimum distance $n-k+1-g$ can exist for $g = 1, 2$ and 3. The length of the code is determined by the number of rational (F_q) points of the curve and the maximum number is actually achieved for many (g, q) pairs.

1. INTRODUCTION.

Codes derived from curves in algebraic geometries have received considerable attention over the past decade. For certain sets of parameters the minimum distance and rate pair lie above the Varshamov-Gilbert bound. The proof of these results requires two deep theorems from algebraic geometry, the Hasse-Weil and Riemann-Roch theorems. It is of interest to consider whether codes can be constructed using similar ideas but without requiring such machinery. After reviewing the basic notions of construction codes from algebraic geometries, the class of hyperelliptic curves is introduced. It will be shown that for such curves the properties of the associated codes can be established by elementary means. The technique however, does not seem to extend to more general classes of curves.

2. AN INTERPRETATION OF REED-SOLOMON CODES.

The classical definition of an extended Reed-Solomon code over the finite field F_q with elements $\alpha_0, \alpha_1, \cdots, \alpha_{q-1}$ is

$$\mathcal{C} = \{(f(\alpha_0), f(\alpha_1), \cdots, f(\alpha_{q-1})),\ f \epsilon L_k\}$$

where

$$L_k = \{f(x) \epsilon F_q[x], \deg f < k\}.$$

The dimension of the code is k and minimum distance $d = n - k + 1$ is the maximum possible. It is of passing interest to note that the maximum length of such codes is q, although it can be extended by other means by one symbol and in fields of characteristic two by two symbols for certain dimensions.

A geometric description of the Reed-Solomon can be pursued as follows. The projective line can be taken as the set of $q+1$ points

$$P^{(1)} = \{(\alpha_i, 1),\ \alpha_i \epsilon F_q\} \cup \{1, 0\}$$

where $Q = (1,0)$ is the point at infinity. The set $\mathcal{L}_k$ is then defined as the collection of rational functions of the form $a(x,y)/b(x,y)$ where $a(x,y)$, $b(x,y)$ are homogeneous polynomials of the same degree with no poles on $P^{(1)}$ except possibly at Q and then of order $< k$. It can then be shown [1] that the set of codewords

$$\mathcal{C} = \{(f(P_1), f(P_2), \cdots, f(P_n)),\ f \epsilon \mathcal{L}_k\}$$

is equivalent to the Reed-Solomon defined earlier.

This approach can be generalized to the more algebraic situation. For a curve X let $P_1, \cdots, P_n$ be the F_q rational points and let D be the formal sum $P_1 + \cdots P_n$ and G be a divisor with disjoint support from D. Let $L(G)$ be the set of rational functions with poles at most a certain order and zeros of at least a certain order for a divisor G. The code is then:

$$C(D,G) = \{(f(P_1), f(P_2), \cdots, f(P_n)),\ f \epsilon L(G)\}.$$

The Riemann-Roch theorem relates the dimension of the space $L(G)$ to a parameter g, the genus of the curve X. The code parameters can be shown to satisfy

$$n = \deg D \quad k = \deg(G) - g + 1 \quad d \geq n - \deg(G).$$

While the code construction is relatively straight forward, the dependence of code parameters on such theorems tends to mask the connection. The following section is an attempt to mimic this development without the use of such theorems.

3. CODES FROM HYPERELLIPTIC CURVES.

The class of algebraic geometry codes derived from hyperelliptic curves over a finite field F_q [2] is considered. The curves contain the set of points $\mathcal{P} = \{(x,y) \mid y^2 + h(x)y = k(x),\ x, y \in F_q\}$ where $h(x)$ is a polynomial of degree at most g and $k(x)$ is a polynomial of degree $2g+1$, both polynomials in $F_q[x]$. This class of curves contains the class of elliptic curves. Notice that if $(x,y) \in \mathcal{P}$ then $(x, -y - h(x)) \in \mathcal{P}$. The size $\mid \mathcal{P} \mid = n$ can be bounded by the Hasse-Weil theorem:

$$\mid n - (q+1) \mid \leq 2g\sqrt{q}.$$

For an algebraic geometry code, one would define a set of rational functions on the curve and proceed to use the Riemann-Roch theorem, [3], to establish bounds on the minimum distance of the code. Our approach here will be to attempt to use the hyperelliptic curves and define a set of bivariate polynomials to define the code and construct an argument that establishes a lower bound on its minimum distance using only elementary algebra. We proceed as follows.

Consider the vector space of polynomials

$$\mathcal{L} = \{f(x) + yg(x) \mid \deg f \leq i,\ \deg g \leq j,\ f, g \in F_q[x]\},$$

a vector space of dimension $i+j+2$ over F_q. We *define* a code

$$\mathcal{C} = \{(l(P_1), \cdots, l(P_n)),\ l \in \mathcal{L},\ P_i \in \mathcal{P}\}$$

The code clearly has length n, dimension $k = i+j+2$ and the only question is its minimum distance. This will be governed by the maximum number of zeros the bivariate polynomials of $\mathcal{L}$ have on the n points of the curve, which generally requires the use of the Riemann-Roch theorem.

However, consider the polynomial

$$a(x) = f^2(x) + f(x)g(x)h(x) - k(x)g^2(x).$$

and note that if $(x_0, y_0) \in \mathcal{P}$ then $f(x_0) + y_0 g(x_0) = 0$ and $a(x_0) = 0$. Under certain conditions the converse is also true, namely that if there exists a point on the curve with x-coordinate x_0, such that $a(x_0) = 0$, then there is a point $(x_0, y_0) \in \mathcal{P}$ which is a zero of $f(x) + yg(x)$. As a polynomial in one variable, the number of its zeros may be bounded by its degree. The degree of $a(x)$ is

$$\eta = \max\{2i, i+j+g, 2j+2g+1\}$$

and for a given g one obtains the best code by minimizing η. The following three theorems can be established from this minimization process:

THEOREM 1: For $g = 1$ the optimum choices of i and j are: (i) if $k = 2m$ then $i = m$ and $j = m-2$ and $\eta = 2m$, (ii) if $k = 2m+1$ then $i = m$ and $j = m-1$ and $\eta = 2m+1$ and in both cases the resulting code is an $(n, k, d \geq n-k)$ code.

THEOREM 2: For $g = 2$ the optimum choices of i and j are: (i) if $k = 2m$ then $i = m$ and $j = m-2$ and $\eta = 2m+1$, (ii) if $k = 2m+1$ then $i = m+1$ and $j = m-2$ and $\eta = 2m+2$ and in both cases the resulting code is an $(n, k, d \geq n-k-1)$ code.

THEOREM 3: For $g = 3$ the optimum choices of i and j are: (i) if $k = 2m$ then $i = m+1$ and $j = m-3$ and $\eta = 2m+2$, (ii) if $k = 2m+1$ then $i = m+1$ and $j = m-2$ and $\eta = 2m+3$ and in both cases the resulting code is an $(n, k, d \geq n-k-2)$ code.

4. COMMENTS.

As far as is known these families of codes appear to be new. They were established by mimicing a technique drawn from algebraic geometry but without the use of the deep results from that theory. The maximum length n that such codes can have is bounded by the Hasse-Weil theorem mentioned previously. For a given g it will depend on the particular polynomials $h(x)$, $k(x)$ chosen. Similar results can be established for higher values of g and a decoding algorithm for these classes of codes has also been derived.

The technique introduced here does not appear to generalize to more complex curves. As an example of this consider the Hermitian codes [4]. The curve is defined as the F_{q^2} rational points satisfying the equation

$$y^q + y = x^{q+1},$$

a curve of genus$q(q-1)/2$. It is straight forward to show that for any finite field F_{q^2} the curve has q^3+1 points, including the point at infinity, all of which can be obtained from a single point on the curve. It can be shown [4] that the basis polynomials for the associated code $L(mQ)$, for a parameter m are

$$\{x^i y^j \mid 0 \le i,\ 0 \le j \le q-1;\ iq + j(q+1) \le m\}.$$

The resulting code parameters are $n = q^3,\ k = m - g + 1,\ d \ge n - m$. However there appears to be no way to reduce the above basis polynomials to a polynomial in one variable, as was done for the hyperelliptic curves.

REFERENCES.

[1] J.H. van Lint and T.A. Springer, Generalized Reed-Solomon codes from algebraic geometry, *IEEE Transactions on Information Theory*, Vol. IT-33, pp. 305-309, 1987.
[2] N. Koblitz, Hyperelliptic cryptosystems, *Journal of Cryptology*, vol.1(1989), 139-150.
[3] W. Fulton, *Algebraic Curves*, Benjamin, New York, 1969.
[4] H. Stichtenoth, A note on Hermitian codes over $GF(q^2)$, *IEEE Transactions on Information Theory*, Vol. 34, pp. 1345-1348, 1988

Development of Tighter Analytical Bounds on the Error Performance of Convolutional Codes

Peter R. Molnar

Stephen G. Wilson

Department of Electrical Engineering
University of Virginia
Charlottesville, VA 22903

We consider improvements to upper and lower bounds on the performance analysis of convolutional codes with maximum likelihood decoding, with special interest in the low SNR regime, where traditional bounds based on a union bound become weak. Interest in this region is promulgated by applications in concatenated coding approaches, where the inner convolutional code operates at low SNR, and in slow fading channels, where computing an average performance requires accurate bounding at low SNR.

The traditional bounds used for convolutional codes (and block codes for that matter) employ a single-neighbor bound for a lower bound, and a transfer function/union bound method for an upper bound [1,2]. The standard calculation upper bounds the probability of first error event using a union bound. This in turn is a bound on the node error probability, and can be used to upper-bound decoded bit error probability as well. Though asymptotically tight at large SNR, the bounds diverge at low SNR, and indeed the upper bound exceeds one. This is simply due to over-counting of probability mass induced in the union bound.

We report work on tightening both upper and lower bounds on the probability of first error event, P_{fe}, for convolutional codes on the AWGN channel with unquantized metric. We incorporate into the analysis of this probability the joint probability that *two* error events have greater metric than the correct trellis path. This joint probability can be expressed in integral form, then properly upper and lower bounded to preserve overall lower/upper bounds to P_{fe}. The expression attained is a function of SNR, code rate, the individual Hamming distances of the two paths, and the "common" Hamming distance. The procedure is conceptually straight-forward, but requires careful bookkeeping of the various Hamming distances for pairs of error paths. Once this augmented weight spectrum is known, computation of the bounds is simple. Due to space limitations, we must refer readers to [3] for details of the development of analytical bounds on the probability of two error events jointly occuring, as well as methods for counting the various pair events, their individual distances and the common distance to the all-zeros path.

In Figure 1, we illustrate the results of upper and lower bounding of the joint probability of two error events having distances 5 and 6 respectively, as well as common distance 3. This corresponds to the two shortest error events in the trellis of the standard rate 1/2, memory 2 convolutional code. Note that the bounds are quite tight, which is beneficial since a multitude of joint events will be considered in the final upper and lower bound, and it is thereby important to accurately calculate this joint probability.

In Figure 2 we show results of the lower bound calculation for this same code. Choice of any finite set of error events and applying our analysis produces a lower bound, and thus the envelope of these bounds provides a lower bound to P_f. Observe the new lower bound is about 0.8 dB tighter at low SNR than the standard single event lower bound, a significant improvement at lower SNR. Also shown are results for error event probability obtained from simulation of the rate 1/2, memory 2 code.

Finally in Figure 3, we show the upper bound, as modified by the incorporation of intersection probabilities. Unfortunately, our improvement on the transfer function bound is rather negligible, in contrast to the significant tightening in the lower bound. Despite the fact that both upper and lower bounds incorporate corrections involving pairwise events, the lower bound incorporates far more pair combinations than the upper bound, and hence the difference remains significant at low SNR. The difference is not due to error in bounding the pairwise probability (see Figure 1).

References

1. Viterbi, A.J., "Convolutional Codes and Their Performance in Communication Systems," IEEE Trans on Communications Technology, October, 1971.

2. Lin, S., and Costello, D.J., *Error Control Coding: Fundamentals and Applications,* Prentice-Hall, 1983.

3. Molnar, P.R., "Development of Tighter Analytical Bounds for Convolutional Codes," M.S. thesis, University of Virginia, 1992.

Figure 1--Probability of intersection vs. SNR, d0 = 5, d1 = 6, dc = 3

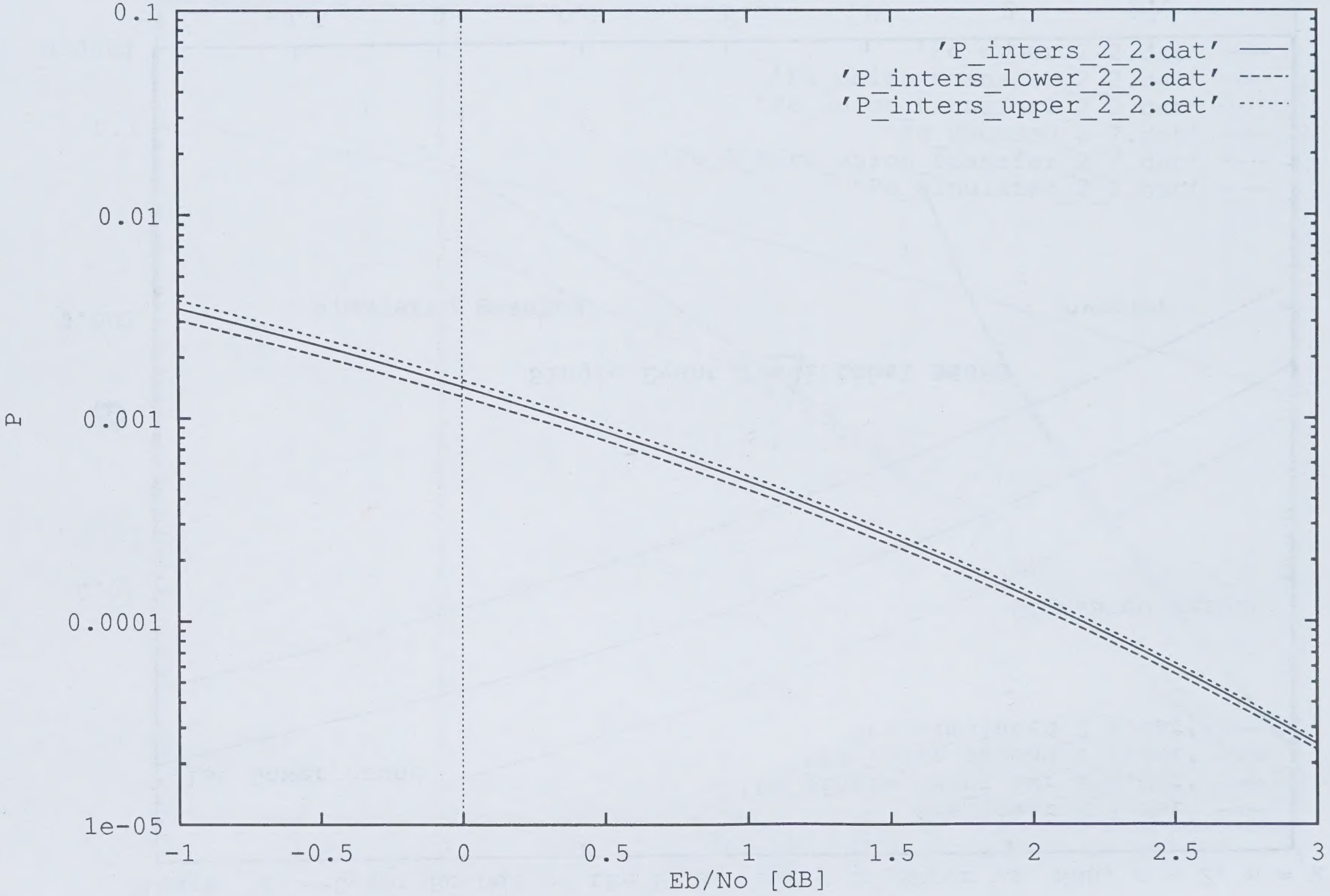


Figure 2 --Lower Bounds on the Probability of Error vs. SNR, m = 2, n = 2

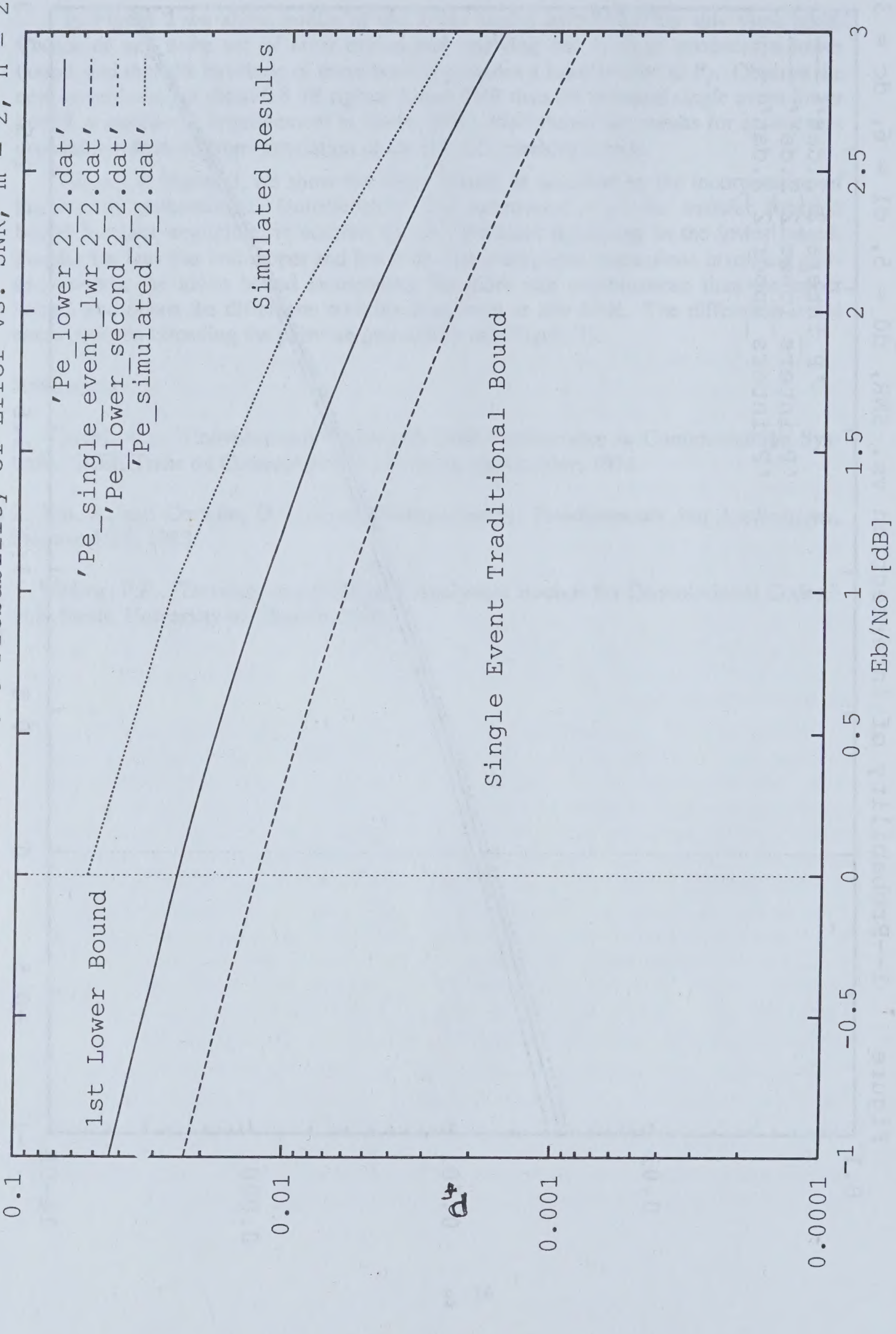
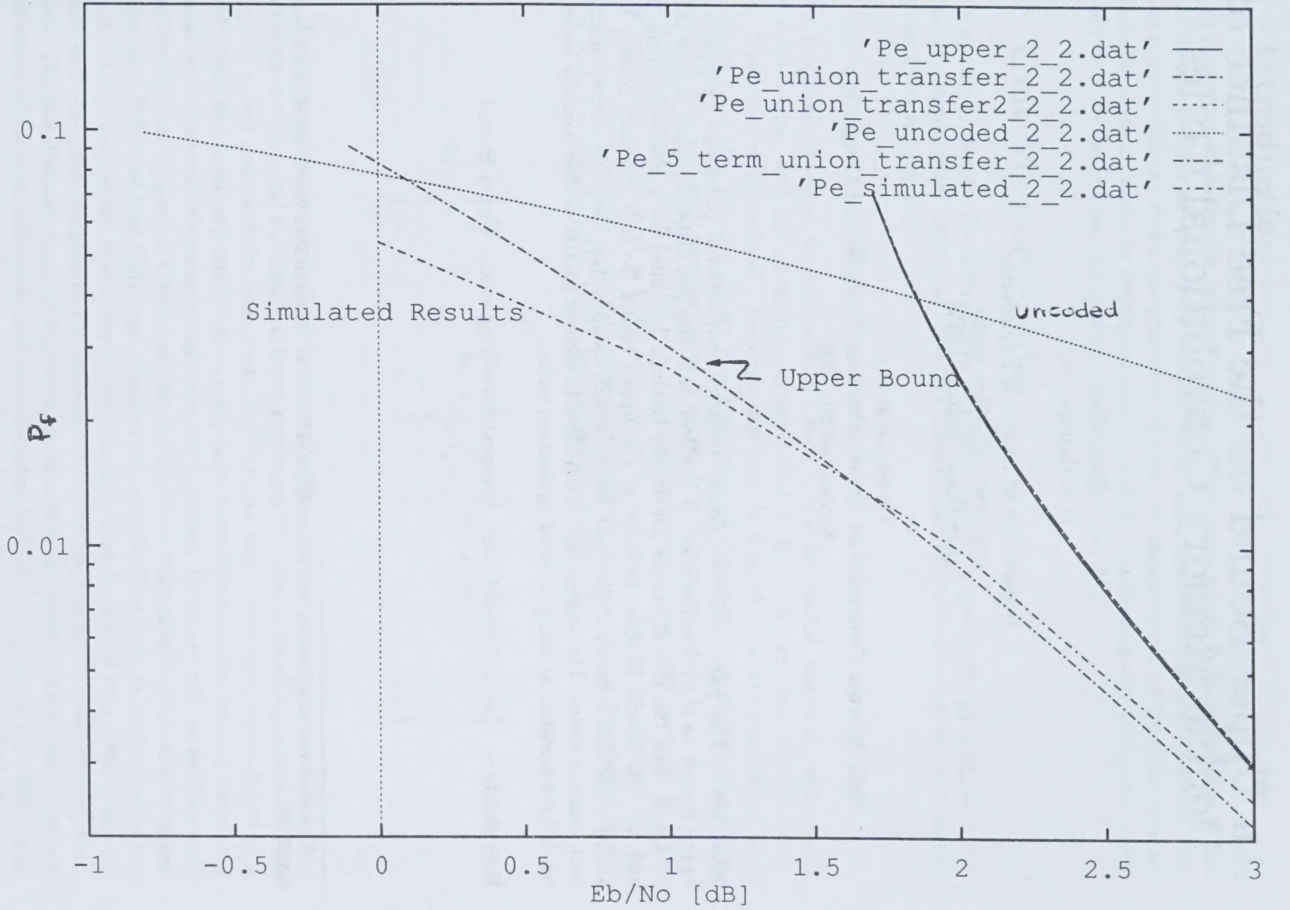


Figure 3 --Upper Bounds on the Probability of Error vs. SNR, m = 2, n = 2



An Upper Bound on the Free Distance of Double Memory Convolutional Codes [1]

D.G. Mills
D.J. Costello, Jr.

Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

Presented at the
1992 Allerton Conference on Communications, Controls, and Computing
September 30, 1992

Abstract —This paper describes double memory convolutional codes and presents an upper bound on their free distance. It is shown that the free distance upper bound can be larger than the free distances previously attained by codes with relatively prime k and n. The bound is often as large as the upper bound for unit memory codes. In addition, double memory codes have lower branch complexities then the corresponding unit memroy codes. The bound can be modified to include partial double memroy codes, whch have poential as unequal error protection codes.

Keywords — Convolutional Codes, Unequal Error Protection, Upper Bound.

---

[1] This work was supported by NSF Grant EID90-17558, NSF Grant NCR89-03429, and NASA Grant NAG5-557.

1 Introduction

This paper describes double memory convolutional codes, which are an extension of the unit memory codes developed by Lee [1]. First, unit memory codes are reviewed, and then double memory codes are presented. Finally, an upper bound on the free distance of double memory codes is developed and examined. Double memory codes are under study as a method of providing unequal error protection [2].

2 Unit Memory Codes

Let u_t and v_t denote the information and code vectors, respectively, at sublock t of a general convolutional code. An (n,k,M) binary convolutional code can be represented by the encoding equation

$$v_t = u_tG_0 + u_{t-1}G_1 + \ldots + u_{t-M}G_M.$$

The information vector for the (n,k,M) code is a k-bit vector, the code vector v_t is an n-bit vector, and the encoding matrices, $G_i, i = 0,1,\ldots M$, are kxn binary matrices. The state complexity of a convolutional code is defined to be the number of state variables,$K = Mk$. (For simplicity, it is assumed that the memory is equally allotted to the input bits, i.e., each encoder input is delayed by M memory units.)

A unit memory code (UMC) is a binary convolutional code with memory $M = 1$. The encoding equation of a UMC is $v_t = u_tG_0 + u_{t-1}G_1$. It can be shown that an (n_o, k_o, m) convolutional code with encoding matrices $g_0, g_1, \ldots, g_m$, is equivalent to the $(n = mn_o, k = mk_o, 1)$ UMC which has the encoding matrices

$$G_0 = \begin{bmatrix} g_0 & g_1 & \cdots & g_{m-1} \\ 0 & g_0 & & g_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & g_0 \end{bmatrix} \quad G_1 = \begin{bmatrix} g_m & 0 & \cdots & 0 \\ g_{m-1} & g_m & & 0 \\ \vdots & & \ddots & \vdots \\ g_1 & \cdots & & g_m \end{bmatrix}$$

The two codes are equivalent in the sense that the output sequences of the two encoders are identical for identical input sequences [1]. The state complexities of both the (n_o, k_o, m) code and the $(mn_o, mk_o, 1)$ UMC are mk_o. The encoding matrices of a UMC that has been converted from a basic code must adhere to the form in the above equation. However, the encoder matrices for a general UMC are not under the same restriction, that is, general UMC matrices are not block triangular matices with constant diagonals. It follows that UMCs are a larger class of codes than basic convolutional codes. In addition, since every basic convolutional code can be converted to a UMC with the same rate and state complexity, the optimal UMC is at least as good as the optimal basic code. The free distance, d_{free}, of a convolutional code is the minimum Hamming distance betweeen all pairs of codewords that are associated with input sequences that differ in at least one subblock. It can be assumed without loss of generality that the first difference between the input sequences appears in subblock 0. Let $v_{t1,t2}$ be the code sequence from time $t1$ to $t2$. For an $(n,k,1)$ UMC, when the only non-zero portion of the information vector is u_0, then the (non-zero) output is $v_{0,1} = u_0[G_0G_1]$. The set of all such u_0's and

$v_{0,1}$'s forms a $(2n, k)$ block code. It follows that the optimal d_{free} of the $(n, k, 1)$ UMC is upperbounded by the minimum Hamming distance of the optimal $(2n, k)$ block code [1]. The optimal block code minimum distances are tabulated in [3]. In several cases, the UMC upper bound is larger than the free distance attained by the optimal codes with the same state complexity for which the greatest common denominator of n_o and k_o is 1 (hereafter called basic codes). Lee conducted an exhaustive search for the optimal UMCs and found several UMCs better than the optimal basic codes,i.e., the rate and complexity of the optimal UMC and basic codes were identical, but the free distance of the optimal UMC was higher. Those results are shown in Table 1.

3 Double Memory Codes

A double memory code (DMC) is a convolutional code with $M = 2$ that can be described by $v_t = u_tG_0 + u_{t-1}G_1 + u_{t-2}G_2$. Any $(n_o, k_o, 2m)$ convolutional code with encoding matrices $g_0, g_1, ..., g_{2m}$ is equivalent to the $(mn_o, mk_o, 2)$ DMC with encoding matrices

$$G_0 = \begin{bmatrix} g_0 & g_1 & \cdots & g_{m-1} \\ 0 & g_0 & & g_{m-2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & g_0 \end{bmatrix} \quad G_1 = \begin{bmatrix} g_m & g_{m+1} & \cdots & g_{2m-1} \\ g_{m-1} & g_m & & g_{2m-2} \\ \vdots & & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_m \end{bmatrix}$$

$$G_2 = \begin{bmatrix} g_{2m} & 0 & \cdots & 0 \\ g_{2m-1} & g_{2m} & & 0 \\ \vdots & & \ddots & \vdots \\ g_{m+1} & \cdots & & g_{2m} \end{bmatrix}.$$

It should be noted that the class of double memory codes, while smaller than the class of UMCs, is larger than the class of basic convolutional codes. This is confirmed by the fact that every basic code can be converted to an equivalent DMC, and every DMC can be converted to an equivalent UMC, but the reverse relationship (UMC to DMC to basic code) does not hold for all UMCs or DMCs.

The state complexity of a $(mn_o, mk_o, 2)$ DMC is $2mk_o$. The free distance is again given by the minimum weight vector resulting from an information sequence that is non-zero in the 0^{th} subblock. For an $(n, k, 2)$ DMC, because the set of such u_0's and their associated outputs can be considered as a $(3n, k)$ block code with $v_{0,2} = u_0[G_0G_1G_2]$, the optimal d_{free} is upper bounded by the highest attainable minimum distance of a $(3n, k)$ block code. The bounds are shown in Table 2. As with UMCs, the free distance of the optimal basic code with the appropriate rate and state complexity provides a lower bound on the optimal DMC free distance.

The bound for $k = 1$ is uninteresting because an $(n, 1, 2)$ basic code is also an $(n, 1, 2)$ DMC. The bound is tight for most values of $k > 1$. However, in some cases, this block code upper bound for the DMC d_{free} is larger than the free distance achieved by the optimal basic code with the same rate and state complexity. In addition, the block code upper bound for DMCs is sometimes as large as the block code upper bound for UMCs, which indicates that free distances for those cases might be as large as those achieved by the optimal UMC. However, the existence of a DMC that attains the block code upper bound is not guaranteed. It should be noted that DMCs have smaller branch complexities than the comparable UMCs, which simplifies decoding.

4 Conclusions

Double memory codes are interesting for several reasons. First, the free distance performance of a DMC is at least as high as the free distance of the basic code of the same rate and state complexity. In addition, the form of DMCs is byte-oriented, which is attractive for use in concatenated coding systems. The simplified decoding methods used for UMCs [5] may be applicable to DMCs after some modifications. Finally, partial double memory codes show potential as unequal error protection codes. These properites of DMCs merit further investigation.

References

[1] L.N. Lee, "Short Unit-Memory Byte-Oriented Binary Convolutional Codes Having Maximal Free Distance," *IEEE Trans. Inf. Theory,* May 1976, pp. 349-352.

[2] D.G. Mills and D.J. Costello,Jr., "Using a Modified Transfer Function to Calculate the Unequal Error Protection Capabilities of Convolutional Codes," submitted to the 1993 IEEE International Symposium on Information Theory.

[3] H.J. Helgert, and R.D. Stinaff, "Minimum-Distance Bounds for Binary Linear Codes," *IEEE Trans. Inf. Theory,* May, 1973, pp. 344-356.

[4] S. Lin and D.J. Costello, Jr., *Error Control Coding: Fundamentals and Applications,* Prentice-Hall,Inc., 1983.

[5] J. Justesen, "Bounded Distance Decoding of Unit Memory Codes," Report IT-121, Institute of Circuit Theory and Telecommunication, Technical University of Denmark, September, 1990.

(n,k)	UMC upper bound	UMC achieved	basic achieved
(4,2)	5	5	5
(6,3)	6	6	6
(8,4)	8	8	7
(10,5)	9	9	8
(12,6)	10	10	10
(6,2)	8	8	8
(9,3)	10	10	10
(12,4)	12	12	12
(15,5)	15	15	13
(18,6)	16	16	15
(8,2)	10	10	10
(12,3)	13	13	13
(16,4)	16	16	16
(20,5)	20	20	18
(24,6)	24	24	20

Table 1

(n_o,k_o)	(n,k)	K	optimal basic	DMC bound	optimal DMC	UMC bound	optimal UMC
(2,1)	(4,2)	4	7	8		8	8
	(6,3)	6	10	10		10	10
	(8,4)	8	12	12		14	
	(10,5)	10	14	15		16	
	(12,6)	12	16	16		19	
(3,1)	(6,2)	4	12	12		12	12
	(9,3)	6	15	15		16	16
	(12,4)	8	18	18		22	
	(15,5)	10	22	22		27	
	(18,6)	12	24	26		32	
(4,1)	(8,2)	4	16	16		16	
	(12,3)	6	20	20		24	
	(16,4)	8	24	24		30	
	(20,5)	10	29	30		36	
	(24,6)	12	33	35		44	
(3,2)	(6,4)	8		8		9	
	(9,6)	12		12		13	
	(12,8)	16		16		16	

Table 2

THE ERROR PROBABILITY OF EXPANSION CODES

ALI S. KHAYRALLAH

ELECTRICAL ENGINEERING DEPARTMENT
UNIVERSITY OF DELAWARE
NEWARK, DE 19716

ABSTRACT

We investigate the behavior of expansion codes in the presence of noise. Expansion codes are a class of error control codes with the property that $d(\mathbf{y},\mathbf{y}') - d(\mathbf{x},\mathbf{x}') \geq d - 1$, where d is the minimum distance of the code, and $(\mathbf{x},\mathbf{y})$, $(\mathbf{x}',\mathbf{y}')$ are distinct input-codeword pairs. We show that for a bounded distance decoder, channel errors cause user errors (from source to sink) in a well controlled manner. We also find an upper bound to the user per-letter error probability in terms of the code parameters. We give several examples of expansion codes and their error probability.

INTRODUCTION

Consider the problem of transmitting information over a discrete memoryless noisy channel. For simplicity we assume a q-ary symmetric channel with crossover probability $\varepsilon/(q-1)$, $0 \leq \varepsilon \leq (q-1)/q$. The relevant distance is the Hamming distance $d(\mathbf{u},\mathbf{v})$, where $\mathbf{u}$ and $\mathbf{v}$ are blocks of the same length. Also let the Hamming weight be $w(\mathbf{u}) = d(\mathbf{u},\mathbf{0})$. In this paper we focus on block error-control codes. An $(n,k;d)$ code has input length k, output length n, and minimum distance d. The encoder map is a one-to-one and onto map

$$f_e: A^k \rightarrow C \subset A^n$$

where A is the alphabet of size q and C is the codebook. We refer to this familiar code as a minimum distance code. In a typical treatment of error-control coding, the encoder map itself is given little attention. However for the subclass of expansion codes to be introduced next, f_e is of crucial importance. We define the set of input-codeword pairs

$$B \overset{\Delta}{=} \{(\mathbf{x},\mathbf{y}):\ \mathbf{x} \in A^k,\ \mathbf{y} = f_e(\mathbf{x}) \in C\}$$

This set emphasizes the role of f_e and is useful in later definitions.

An $(n,k;d)$ code is an expansion code [4] if it satisfies the expansion property

$$\forall\ (\mathbf{x},\mathbf{y}),\ (\mathbf{x}',\mathbf{y}') \in B \text{ such that } \mathbf{x} \neq \mathbf{x}',\ d(\mathbf{y},\mathbf{y}') \geq d(\mathbf{x},\mathbf{x}') + d - 1 \qquad (1.a)$$

Since the code has minimum distance d, it follows that

$$\exists\ (\mathbf{x},\mathbf{y}),\ (\mathbf{x}',\mathbf{y}') \in B \text{ such that } d(\mathbf{x},\mathbf{x}') = 1 \text{ and } d(\mathbf{y},\mathbf{y}') = d \qquad (1.b)$$

If the code is linear, then (1) becomes

$$\forall\ (\mathbf{x},\mathbf{y}) \in B \ \text{ such that } \ \mathbf{x} \neq \mathbf{0},\ w(\mathbf{y}) \geq w(\mathbf{x}) + d - 1 \tag{2.a}$$

$$\exists\ (\mathbf{x},\mathbf{y}) \in B \ \text{ such that } \ w(\mathbf{x}) = 1 \ \text{ and } \ w(\mathbf{y}) = d \tag{2.b}$$

One can see from (1) or (2) that an expansion code preserves and expands the distances among inputs. Hence it is most useful when input distance is meaningful. This is the case for instance when the code is a stage in a cascade of codes. It is also most meaningful in situations where error probabilities need not be extremely small. The price paid for the expansion property is a decrease in coding rate. Performance bounds similar to the Gilbert and Plotkin bounds ([1], [6]) are derived in [4]. Trivial examples of expansion codes are $(n,1;1)$ repetition codes. Also, given the generator matrix $\mathbf{G}$ of an $(n,k;d)$ minimum distance code, one can obtain an $(n+k,k;d+1)$ systematic expansion code with generator matrix $\mathbf{G}' = [\mathbf{I}\ \mathbf{G}]$, where $\mathbf{I}$ is the identity matrix. In general, this simple construction does not yield the shortest codes.

CHANNEL ERRORS AND USER ERRORS

In this paper we investigate the behavior of expansion codes in the presence of noise. Using Fig. 1 below, we refer to errors between the transmitted codeword $\mathbf{y}$ and the received word $\mathbf{z}$ as channel errors, as usual. We refer to errors between the encoder input $\mathbf{x}$ and the decoder output $\hat{\mathbf{x}}$ as user errors. Our premise is that the users (source and sink) are concerned with user errors. In particular, they would like $p(\hat{\mathbf{x}}|\mathbf{x})$ to decrease with $d(\mathbf{x},\hat{\mathbf{x}})$, thus making large errors unlikely. The relevant average error measure is the user per-letter probability

$$P_{el}^{x} \overset{\Delta}{=} \frac{1}{k}\sum_{i=0}^{k-1} P(X_i \neq \hat{X}_i) \tag{3}$$

In general, the effect of channel errors on user errors is obscure, and (3) is difficult to compute. For expansion codes however, we are able to show precisely how channel errors cause user errors. Also, we find an interesting expression for (3) in terms of the familiar word error probability and the code per-letter error probability.

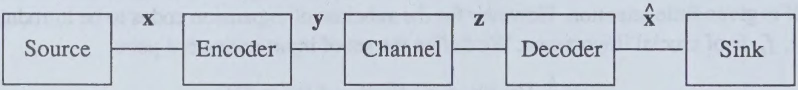


Fig.1 Error-control code components

We consider an $(n,k;d)$ expansion code, and we restrict our attention to bounded distance decoders [2]. The decoding radius is ρ, $0 \leq \rho \leq \lfloor (d-1)/2 \rfloor$, and the decoding sets are the spheres

$$S_{\mathbf{y}} = \{\mathbf{z}\text{: } d(\mathbf{z},\mathbf{y}) \leq \rho\}$$

The volume of each sphere is given by

$$s_\rho \overset{\Delta}{=} \sum_{0 \le w \le \rho} \binom{n}{w} (q-1)^w$$

The total decoding region is

$$S = \bigcup_{\mathbf{y} \in C} S_{\mathbf{y}}$$

The decoder f_d can be described in two stages. The first stage is a decision rule

$$f_{d1}: A^n \to C \cup \{?\}$$

$$f_{d1}(\mathbf{z}) = \begin{cases} \hat{\mathbf{y}} & \mathbf{z} \in S_{\hat{\mathbf{y}}} \\ ? & \mathbf{z} \notin S \end{cases}$$

where ? denotes a detected (and uncorrected) error.

We will assume that in the event of a detected error, the receiver will ask for a retransmission, according to some ARQ scheme [5]. We also assume that $\mathbf{z}$ will eventually fall in S, and the first stage will produce an estimate of the transmitted codeword. The second stage of the decoder is simply the inverse of the encoder map

$$f_{d2}: C \to A^k$$

$$f_{d2}(\mathbf{y}) = f_e^{-1}(\mathbf{y})$$

We now discuss the effect of channel error on decoded input errors. This will highlight the predictable manner in which expansion codes handle errors. Suppose codeword $\mathbf{y}$ is sent over the channel, and some word $\mathbf{z} \in S_{\hat{\mathbf{y}}}$ is received, where $d(\mathbf{z},\mathbf{y}) = v$, $d(\mathbf{y},\hat{\mathbf{y}}) = u$, and $d(\mathbf{z},\hat{\mathbf{y}}) = w$. This is illustrated in Fig. 2 below.

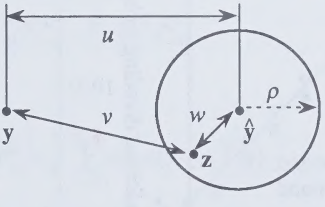


Fig. 2 Error event in a bounded distance decoder

The probability of receiving $\mathbf{z}$ given $\mathbf{y}$ was sent is

$$p(\mathbf{z}|\mathbf{y}) = \left(\frac{\varepsilon}{q-1}\right)^v (1-\varepsilon)^{n-v} \overset{\Delta}{=} \pi_v \tag{4}$$

which decreases with v for $\varepsilon \le (q-1)/q$. The probability of decoding $\hat{\mathbf{y}}$ given that $\mathbf{y}$ was sent also depends only on u. That is,

$$p(\hat{\mathbf{y}}|\mathbf{y}) = P(\mathbf{z} \in S_{\hat{\mathbf{y}}} \mid \mathbf{y}) \overset{\Delta}{=} \Pi_u \tag{5}$$

While we cannot say that Π_u decreases with u, we can show that it is lower and upper bounded by constants times $\pi_{u-\rho}$.

Proposition 1

The conditional decoding probability Π_u is upper and lower bounded by

$$\binom{d}{\rho} \pi_{u-\rho} \le \Pi_u \le s_\rho \, \pi_{u-\rho} \tag{6}$$

for $d \le u \le n$.

The proof of Proposition 1 along with tighter bounds to Π_u is left to the appendix. We can now say that Π_u is "approximately proportional" to $\pi_{u-\rho}$, and we write this as

$$\Pi_u \propto \pi_{u-\rho}$$

Moreover, since $\pi_{u-\rho}$ decreases with u, then Π_u approximately decreases with u. The result of Proposition 1 is illustrated in Fig. 3 below.

The decoder output is $\hat{\mathbf{x}} = f_{d2}(\hat{\mathbf{y}})$ with $d(\mathbf{x},\hat{\mathbf{x}}) \le u - d + 1$. Now $w \le \rho$, and by the triangle inequality $u - \rho \le v \le u + \rho$. Thus at worst, v channel errors can cause no more than $v + \rho - d + 1$ user errors, and at best, they can cause $v - \rho - d + 1$ or fewer user errors. While we cannot say precisely that $p(\hat{\mathbf{x}}|\mathbf{x})$ decreases with $d(\mathbf{x},\hat{\mathbf{x}})$, the previous discussion indicates that this is essentially true. In particular, for the case $d(\mathbf{x},\hat{\mathbf{x}}) = d(\mathbf{y},\hat{\mathbf{y}}) - d + 1$, we have $p(\hat{\mathbf{x}}|\mathbf{x}) = \Pi_{d(\mathbf{x},\hat{\mathbf{x}})+d-1} \propto \pi_{d(\mathbf{x},\hat{\mathbf{x}})+d-1-\rho}$.

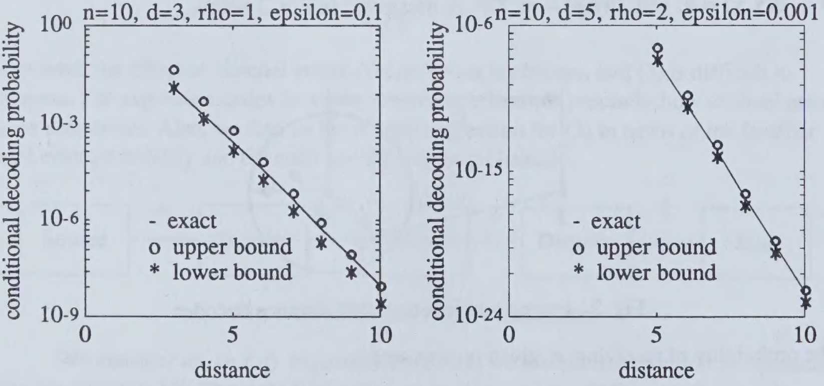


Fig. 3 Π_u and its bounds v/s u

For a linear code, let a_i denote the number of codewords of weight i, $0 \le i \le n$. The familiar word error probability can be written as

$$P_{ew} = q^{-k} \sum_{\substack{\mathbf{y},\hat{\mathbf{y}} \in C \\ \mathbf{y} \neq \hat{\mathbf{y}}}} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} \tag{7}$$

$$= \sum_{i=1}^{n} a_i \, \Pi_i \tag{8}$$

where (8) holds for linear codes, and the probability of correct decoding can be written as

$$P_c = \Pi_0$$

We also define the code per-letter error probability which will be used in the next result.

$$P^y_{el} \triangleq \frac{1}{n} \sum_{i=0}^{n-1} P(Y_i \neq \hat{Y}_i)$$

$$= \frac{1}{n} q^{-k} \sum_{\mathbf{y},\hat{\mathbf{y}} \in C} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} \, d(\mathbf{y},\hat{\mathbf{y}}) \tag{9}$$

$$= \frac{1}{n} \sum_{i=0}^{n} a_i \, \Pi_i \, i \tag{10}$$

where (10) holds for linear codes. The idea of a code per-letter probability is somewhat unusual, and it might be useful in other situations. We note that Proposition 1 may be related to the approximation $P^x_{el} \cong (1 - P_c)\, d/n$ mentioned in [3].

Proposition 2

The user per-letter probability is bounded from above by

$$P^x_{el} \leq \frac{q^{-k}}{k} \sum_{\substack{\mathbf{y},\hat{\mathbf{y}} \in C \\ \mathbf{y} \neq \hat{\mathbf{y}}}} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} \left(d(\mathbf{y},\hat{\mathbf{y}}) - d + 1\right)$$

$$= \frac{n}{k} P^y_{el} - \frac{d-1}{k} P_{ew}$$

Proof

$$P^x_{el} = \frac{1}{k} \sum_{i=0}^{k-1} E\{d(X_i \neq \hat{X}_i)\}$$

$$= \frac{1}{k} E\{d(\mathbf{X},\hat{\mathbf{X}})\}$$

$$= \frac{1}{k} \sum_{\mathbf{x},\hat{\mathbf{x}}} p(\mathbf{x})\, p(\hat{\mathbf{x}}|\mathbf{x})\, d(\mathbf{x},\hat{\mathbf{x}})$$

where

$$p(\mathbf{x}) = p(\mathbf{y}) = q^{-k}$$

and

$$p(\hat{\mathbf{x}}|\mathbf{x}) = p(\hat{\mathbf{y}}|\mathbf{y}) = \Pi_{d(\mathbf{y},\hat{\mathbf{y}})}$$

where $(\mathbf{x},\mathbf{y})$, $(\hat{\mathbf{x}},\hat{\mathbf{y}}) \in B$. Moreover, using (1.a) and (1.b),

$$\begin{aligned} P_{el}^{x} &\le \frac{1}{k} q^{-k} \sum_{\substack{\mathbf{y},\hat{\mathbf{y}} \in C \\ \mathbf{y} \ne \hat{\mathbf{y}}}} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} \left(d(\mathbf{y},\hat{\mathbf{y}}) - d + 1\right) \\ &= \frac{1}{k} q^{-k} \sum_{\mathbf{y},\hat{\mathbf{y}} \in C} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} d(\mathbf{y},\hat{\mathbf{y}}) - \frac{1}{k} q^{-k} \sum_{\substack{\mathbf{y},\hat{\mathbf{y}} \in C \\ \mathbf{y} \ne \hat{\mathbf{y}}}} \Pi_{d(\mathbf{y},\hat{\mathbf{y}})} (d - 1) \\ &= \frac{n}{k} P_{el}^{y} - \frac{d-1}{k} P_{ew} \end{aligned}$$

The result of Proposition 2 can be restated as a relation between the expected user distance and the expected code distance.

Proposition 3

$$E\{d(\mathbf{X},\hat{\mathbf{X}}) \mid \mathbf{X} \ne \hat{\mathbf{X}}\} \le E\{d(\mathbf{Y},\hat{\mathbf{Y}}) \mid \mathbf{Y} \ne \hat{\mathbf{Y}}\} - (d-1)$$

Proof

$$\begin{aligned} E\{d(\mathbf{X},\hat{\mathbf{X}}) \mid \mathbf{X} \ne \hat{\mathbf{X}}\} &= q^{-k} \sum_{\mathbf{x},\hat{\mathbf{x}} \ne \mathbf{x}} \frac{p(\hat{\mathbf{x}}|\mathbf{x})}{P_{ew}} d(\mathbf{x},\hat{\mathbf{x}}) \\ &= q^{-k} \sum_{\mathbf{y},\hat{\mathbf{y}} \ne \mathbf{y}} \frac{\Pi_{d(\mathbf{y},\hat{\mathbf{y}})}}{P_{ew}} \left(d(\mathbf{y},\hat{\mathbf{y}}) - d + 1\right) \\ &= E\{d(\mathbf{Y},\hat{\mathbf{Y}}) \mid \mathbf{Y} \ne \hat{\mathbf{Y}}\} - (d - 1) \end{aligned}$$

Proposition 3 can be viewed as a weak (average) form of expansion property. It can be interpreted as follows: Given that a user error occurred, the expected number of errors in the decoded input $\hat{\mathbf{x}}$ is bounded from above by the expected number of errors in the decoded codeword $\hat{\mathbf{y}}$ minus the distance enhancement $(d-1)$.

Remark

The inequalities in Propositions 2 and 3 become equalities for exact expansion codes, i.e. those for which (1.b) holds with equality for all $(\mathbf{x},\mathbf{y})$.

EXAMPLES

We give several examples of binary linear expansion codes found by computer search. The generator matrix is denoted $\mathbf{G}_{(n,k;d)}$, and its vector of weight coefficients is denoted $\mathbf{a}_{(n,k;d)}$. As a point of reference, we also give $n_{(k;d)}$, the shortest output length of any minimum distance linear code with parameters k and d, which can be found in [7]. For each code we plot P_{el}^{x}, P_{ew}, P_{ew}/k, and the upper bound to P_{el}^{x} from Proposition 2. Note that for small values of ε, $P_{el}^{x} \cong P_{ew}/k$, since the dominant error events are those that produce $\hat{\mathbf{y}}$ with $d(\mathbf{y},\hat{\mathbf{y}}) = d$, which correspond to $\hat{\mathbf{x}}$ with $d(\mathbf{x},\hat{\mathbf{x}}) = 1$.

$$\mathbf{G}_{(11,5;3)} = \begin{bmatrix} 00000000111 \\ 00000011100 \\ 00001110000 \\ 00111000000 \\ 11100000000 \end{bmatrix}$$

$$\mathbf{a}_{(11,5;3)} = (1\ 0\ 0\ 5\ 4\ 3\ 8\ 7\ 3\ 1\ 0\ 0)$$

$$n_{(5;3)} = 9$$

$$\mathbf{G}_{(13,5;4)} = \begin{bmatrix} 0000000001111 \\ 0000001111000 \\ 0000111000100 \\ 0011101100000 \\ 1110110000000 \end{bmatrix}$$

$$\mathbf{a}_{(13,5;4)} = (1\ 0\ 0\ 0\ 3\ 5\ 7\ 7\ 4\ 3\ 1\ 1\ 0\ 0)$$

$$n_{(5;4)} = 10$$

$$\mathbf{G}_{(16,6;5)} = \begin{bmatrix} 0000000000011111 \\ 0000000011111000 \\ 0000001110010100 \\ 0000111011000000 \\ 0011101100000000 \\ 1110110000000000 \end{bmatrix}$$

$$\mathbf{a}_{(16,6;5)} = (1\ 0\ 0\ 0\ 0\ 6\ 9\ 12\ 12\ 6\ 7\ 8\ 3\ 0\ 0\ 0\ 0)$$

$$n_{(6;5)} = 14$$

$$\mathbf{G}_{(18,7;5)} = \begin{bmatrix} 000000000000011111 \\ 000000000011111000 \\ 000000001110010100 \\ 000000111011000000 \\ 000011101100000000 \\ 001110110000000000 \\ 111011000000000000 \end{bmatrix}$$

$$\mathbf{a}_{(18,7;5)} = (1\ 0\ 0\ 0\ 0\ 7\ 11\ 16\ 20\ 17\ 17\ 19\ 15\ 4\ 0\ 1\ 0\ 0\ 0)$$

$$n_{(7;5)} = 15$$

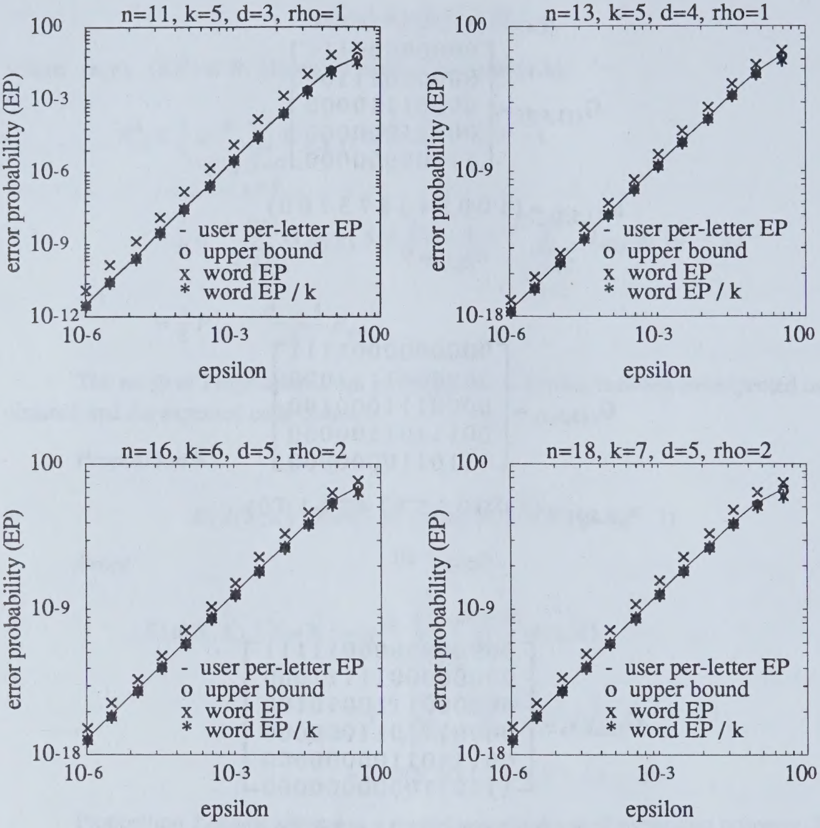


Fig. 4 P^{x}_{el}, P_{ew}, P_{ew} / k, and the upper bound to P^{x}_{el} (Proposition 2)

APPENDIX: THE CONDITIONAL DECODING PROBABILITY

Without loss of generality suppose codeword $\mathbf{0}$ is sent over the channel and codeword $\hat{\mathbf{y}}$ of weight u is decoded. (See Fig. 2 with $\mathbf{y} = \mathbf{0}$.) The number $N(u,v,w)$ of channel outputs $\mathbf{z}$ of weight v at distance w from $\hat{\mathbf{y}}$ is given by [2]

$$N(u,v,w) = \sum_{\substack{i_1,i_2,i_3 \\ i_1+i_2+i_3=w \\ u-i_2+i_3=v}} \binom{u}{i_1} (q-2)^{i_1} \binom{u-i_1}{i_2} \binom{n-u}{i_3} (q-1)^{i_3}$$

where $\binom{i}{j} = 0$ if $i < 0$, $j < 0$, or $j > i$. We now write the conditional decoding probability Π_u in terms of $N(u,v,w)$ and π_v

$$\Pi_u = \sum_{\substack{0 \le w \le \rho \\ u-\rho \le v \le u+\rho}} N(u,v,w)\ \pi_v$$

The next result makes use of the fact that π_v decreases with v.

Proposition 4

The conditional decoding probability Π_u is bounded from below by

$$\Pi_u \ge \sum_{0 \le w \le \rho} \left(\binom{u}{w} \pi_{u-w} + \binom{n-u}{w} (q-1)^w \pi_{u+w} \right) \tag{11}$$

which simplifies to a looser lower bound

$$\Pi_u \ge \binom{u}{\rho} \pi_{u-\rho} \ge \binom{d}{\rho} \pi_{u-\rho} \tag{12}$$

It is bounded from above by

$$\Pi_u \le \sum_{0 \le w \le \rho} \binom{n}{w} (q-1)^w \pi_{u-w} \tag{13}$$

which simplifies to a looser upper bound

$$\Pi_u \le s_\rho\, \pi_{u-\rho} \tag{14}$$

Proof

Note that by the triangle inequality, $u-w \le v \le u+w$, hence $N(u,v,w) = 0$ if $v < u-w$ or $v < u+w$. It is easy to show that

$$N(u,u-w,w) = \binom{u}{w} \tag{15}$$

$$N(u,u+w,w) = \binom{n-u}{w} (q-1)^w \tag{16}$$

where (15) and (16) are the number of words of minimal weight and maximal weight at distance w from $\hat{y}$, respectively. Thus

$$\Pi_u = \sum_{0 \le w \le \rho} \ \sum_{u-w \le v \le u+w} N(u,v,w)\ \pi_v$$

$$\ge \sum_{0 \le w \le \rho} \binom{u}{w} \pi_{u-w} + \binom{n-u}{w} (q-1)^w \pi_{u+w}$$

which proves (11). Since π_i decreases with its index i, we take the term in the summation with the lowest index, yielding the first inequality in (12). The second inequality follows since $u \ge d$.

To show the upper bounds, note that

$$\sum_{u-w \le v \le u+w} N(u,v,w) = \binom{n}{w} (q-1)^w \tag{17}$$

$$\sum_{\substack{u-w \le v \le u+w \\ 0 \le w \le \rho}} N(u,v,w) = \sum_{0 \le w \le \rho} \binom{n}{w} (q-1)^w = s_\rho \tag{18}$$

where (17) is the number of words at distance w from $\hat{\mathbf{y}}$. Since all words at distance w from a codeword of weight u have weights no less than $u-w$, using (17) we get

$$\Pi_u \le \sum_{0 \le w \le \rho} \binom{n}{w} (q-1)^w \, \pi_{u-w}$$

which shows (13). This can be simplified to a looser upper bound using (18), which shows (14). The result of Proposition 4 holds for any bounded distance decoder. The second inequality in (12) and inequality (14) were stated earlier as Proposition 1.

BIBLIOGRAPHY

[1] Berlekamp, E. R., Algebraic coding theory, McGraw-Hill (1968).

[2] Blahut, R. E., Theory and practice of error control codes, Addison-Wesley (1984).

[3] Clark, G. C. and Cain, J. B., Error-correction coding for digital communications, Plenum Press (1981).

[4] Khayrallah, A. S., "Expansion channel codes: Performance bounds and examples," *Proceedings of The Conference On Information Sciences and Systems* (1992).

[5] Lin, S. and Costello, D.J., Error control coding: Fundamentals and applications, Prentice Hall (1983).

[6] Peterson, W. W. and Weldon, E. R., Error-correcting codes, second edition, MIT Press (1972).

[7] Verhoeff, T., "An updated table of minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, no. 5, pp. 665-680 (1987).

MAXIMUM LIKELIHOOD ERROR-TRAPPING DECODING OF BINARY CYCLIC CODES

Wonjin Sung and John T. Coffey
Electrical Engineering and Computer Science Department,
University of Michigan, Ann Arbor, MI 48109

The error-trapping technique is an effective and simple method to decode (n, k, t) cyclic codes with relatively high redundancy and small error-correctability. The method can be adapted to decode any cyclic code by using an auxiliary set of polynomials called covering polynomials. The method was originally proposed by Kasami [1]; a more recent description is given by Wei [3]. The determination of the optimum set of covering polynomials is research problem 16.12 in MacWilliams and Sloane [2]. Among the attractive features of the algorithm is that it can be used to decode past the guaranteed error correcting power of the code, up to complete hard decision decoding. The method can also be extended easily to soft-decision decoding, where we find the closest soft decision codeword, subject to a maximum number of hard errors. In this work, we classify the cyclic codes of length up to 31 according to the number of covering polynomials, and derive new analytic bounds on the number of covering polynomials required in various circumstances.

We consider the decoding of binary cyclic codes of length 31 or less, including the decoding of error patterns of weight $t+1$ or higher. We classify the error patterns to be trapped as (1) all error patterns of weight τ (important in soft decision decoding), (2) all coset leaders of weight τ, and (3) all unique coset leaders of weight τ. The number of covering polynomials is very much dependent on the *interval patterns* which denote how error bits are spaced in a error vector. Two error patterns are said to have the same interval pattern if they are cyclic shifts of each other. Using a combination of analysis, exhaustive search, computational shortcuts, and a greedy algorithm, we determine the number of covering polynomials needed for codes of length up to 31. We summarize these results in the table that follows. (Due to the limited space, we give only a partial list.) We also provide the number of cosets covered by these covering polynomials, which tells about the improvement on the error performance of the decoder.

We extend Wei's results [3] on the optimal set of monomials to the error patterns of weight $\tau > 3$ for codes that have rates $R < 2/\tau$. For $\tau = 4$, for example, we claim that $\lceil \frac{k}{2p} \rceil$ is the necessary and sufficient number of monomials that are sufficient to trap every error pattern of weight $\tau = 4$, where $p = (n-1)/2 - k$. Similar patterns appear for higher τ. Some interesting behavior can be observed in the tables. There are two equivalent codes (the $(17, 9, 5)$ codes) that exhibit different behavior when the decoding of coset leaders of weight greater than t is considered. Different behavior regarding the decoding of the $(31, 16, 7)$ QR and BCH codes was observed, which showed that the QR code behaved better with same decoder complexity when decoding errors of weight t. Simulation of the performance of soft-decision decoding using covering polynomials was also performed. The results show that approximately $1.5 \sim 2.0$ dB gain is achieved at a bit error rate of 10^{-4} for soft-decision using covering polynomial that trap error pattern of weight $t+1$ or less.

(n,k,d,t) : (codeword length, dimension, minimum distance, error correctability)
G : factors of *G*enerator polynomial in octal form
R : covering *R*adius
Am : *A*ll error patterns of weight $t+m$
Cm : *C*oset leaders of weight $t+m$
Um : *U*nique coset leaders of weight $t+m$ $(m = 0, 1, \cdots, R-t.)$
cst : the number of the *cosets* covered
ip : the number of the *i*nterval *p*atterns to be trapped
cp : the number of the *c*overing *p*olynomials achieved using an ordered Greedy algorithm

			U0			A1		C1			U1			A2		C2			U2			A3		C3			U3		
(n,k,d,t)	G	R	cst	ip	cp	ip	cp	cst	ip	cp	cst	ip	cp	ip	cp	cst	ip	cp	cst	ip	cp	ip	cp	cst	ip	cp	cst	ip	cp
$(15,8,4,1)^*$	3.7.23	4	15	0	1	1	2	60	1	2	15	0	1	16	3	49	14	2	0	0	·	71	11	3	5	3	0	0	·
$(15,6,6,2)^*$	3.23.37	5	105	0	1	4	2	225	4	2	65	1	2	35	3	150	22	3	0	0	·	131	6	16	9	4	0	0	·
$(17,9,5,2)^*$	471	3	136	1	2	19	3	102	9	2	0	0	·	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
$(17,8,6,2)^*$	3.471	5	136	0	1	12	2	238	12	2	0	0	·	84	5	119	48	5	0	0	·	294	11	1	1	2	0	0	·
$(21,15,4,1)^*$	3.7.13	3	21	0	1	5	2	31	5	2	0	0	·	54	11	11	19	8	0	0	·	-	-	-	-	-	-	-	-
$(21,10,5,2)$	7.13.165	6	210	0	1	19	2	1120	16	2	1120	16	2	165	6	630	69	5	0	0	·	759	15	63	37	9	0	0	·
$(21,8,6,2)$	3.127.165	6	210	0	1	4	2	1135	4	2	945	1	2	71	2	3255	61	2	1890	16	2	476	4	2940	329	5	0	0	·
$(23,11,8,3)$	3.5343	7	1771	22	2	220	7	1771	220	7	0	0	·	1133	16	253	177	15	0	0	·								
$(27,6,6,2)$	3.7.1001001	12	351	0	1	0	1	2838	0	1	2754	0	1	0	1	15687	0	1	14094	0	1	7	2	61695	6	2	47385	4	2
$(31,25,4,1)^*$	3.73	3	31	0	1	10	3	31	10	3	0	0	·	135	27	1	5	6	0	0	·	-	-	-	-	-	-	-	-
$(31,15,8,3)$	3.45.57.73	6	4495	40	2	560	9	17515	560	9	4340	85	4																

* The number of sufficient covering polynomials to trap all error patterns of weight up to the covering radii is listed.

References

[1] T. Kasami, "A decoding procedure for multiple-error-correcting cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-10, pp. 134–138, 1964.

[2] F. J. MacWilliams and N. J. A. Sloan, *The Theory of Error-Correcting Codes*, North-Holland, New York, 1977.

[3] V. K. Wei, "An error-trapping decoder for nonbinary cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 538–541, 1984.

GENERAL STRUCTURAL SUBBAND DECOMPOSITION OF ADAPTIVE FILTERS FOR SYSTEM IDENTIFICATION PROBLEMS

S. SONG, M. R. PETRAGLIA, and S. K. MITRA
University of California, Santa Barbara, CA, and

A. MAHALANOBIS
Martin Marietta Electronic Systems, Orlando, FL

Abstract

The theory of structural subband decomposition is extended to adaptive FIR filters for system identification problems. It is shown that the subband decomposition is equivalent to transformation of input data by orthogonal matrices of which the Walsh-Hadamard Transform (WHT) is a special case. The proposed method is a generalization of the WHT-based transform domain adaptive filtering which is known to enhance the convergence speed of adaptive filters. It is also shown that the interpolator network of the subband structure is a fast implementation of the required transform and is therefore attractive from a practical standpoint. The convergence properties of subband adaptive filters are studied using the LMS algorithm for system identification application. The results demonstrate the advantages of the proposed structure compared to conventional adaptive filters.

INTRODUCTION

Various methods have been proposed to reduce the computational complexity and increase the convergence speed of the adaptive filtering algorithms. The *least mean squares* (LMS) algorithm is popular for its low computational complexity and memory requirement, while the recursive least squares (RLS) algorithm is generally known for its higher convergence speed. The convergence speed of the LMS algorithm is governed by the spread of eigenvalues of the auto-correlation matrix of the input data (assuming that the data is independent from iteration to iteration), and by reducing the eigenvalue spread, the convergence speed can be enhanced. To this end, various transform-domain adaptive filtering (TDAF) methods have been devised [1]-[5]. The idea behind TDAF is to pre-whiten the input data using unitary transforms. We limit our study to the short term whitening cases only. The best transform for this purpose is the well-known Karhunen-Loeve Transform (KLT). However, KLT is difficult to compute and generally is replaced by simpler transforms such as the Discrete Fourier Transform (DFT). Marshall et. al. [4] compared various unitary transforms for convergence enhancement. They showed that a relatively simple transform such as the Walsh-Hadamard Transform (WHT) offers good performance, although no absolute "winner" among the transforms can be identified. They also proposed the use of a new simpler transform called the "power-of-two (PO2)" and examined its performance for the 8×8 case but did not generalize it further. Among the simpler transforms, the WHT is computationally attractive since its elements are either 1 or -1. The real shortcoming of the WHT is that the filter tap length must be a power of 2.

Recently the concept of structural subband decomposition of FIR filters was advanced by Mitra and Mahalanobis, and applied to the efficient design and implementation of such filters [6]. Their proposed structure can also be used in adaptive filtering. The traditional TDAF using WHT results as a special case when the filter length is $N = 2^L$, and the full N band decomposition is applied. Using the theory of structural subband decomposition, we generalize in this paper the WHT-based TDAF method.

MATRIX REPRESENTATION OF SUBBAND FIR STRUCTURE

The M-branch structural subband decomposition of an FIR filter H(z) of length N = MP with M $= 2^L$ is given by [6]:

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} = \begin{bmatrix} 1 & z^{-1} & \dots & z^{-(M-1)} \end{bmatrix} \mathbf{R}_M \begin{bmatrix} G_0(z^M) \\ G_1(z^M) \\ \vdots \\ G_{M-1}(z^M) \end{bmatrix} \qquad (1)$$

where $\mathbf{R}_M$ is an M×M Hadamard matrix and $G_k(z) = \sum_{n=0}^{P-1} g_k(n) z^{-n}$ is an FIR filter of length P. Note that $\mathbf{R}_M$ can be expressed as an L-fold Kronecker product using the equation

$$\mathbf{R}_M = \underbrace{\mathbf{R}_2 \otimes \mathbf{R}_2 \otimes \cdots \otimes \mathbf{R}_2}_{L-\text{times}} \tag{2}$$

with $\mathbf{R}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. The decomposition of Eq. (1) can be alternately written as:

$$H(z) = \sum_{k=0}^{M-1} I_k(z) G_k(z^M), \tag{3}$$

where $I_k(z)$ is called the interpolator and $G_k(z^M)$ is called the subfilter of the k-th branch.

In matrix notation, a general M-band decomposition can be written as

$$\mathbf{h} = \sum_{k=0}^{M-1} \mathbf{A}_k \mathbf{g}_k, \tag{4}$$

where $\mathbf{h} = [h(0)\ \ h(1)\ \ h(2)\ \ldots\ h(N-1)]^t$ is the filter vector of length N, and the vector $\mathbf{g}_k = [g_k(0)\ \ g_k(1)\ \ g_k(2) \ldots g_k(P-1)]^t$ contains the P non-zero coefficients of the k-th subfilter. The N×P subband transform matrices $\mathbf{A}_k$ are determined by the interpolator structures. To this end, we rewrite Eq. (4) in the form

$$\mathbf{h} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \cdots & \mathbf{A}_{M-1} \end{bmatrix} \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{M-1} \end{bmatrix} = \mathbf{T}_M \mathbf{f}_M, \tag{5}$$

where $\mathbf{T}_M = [\mathbf{A}_0\ \ \mathbf{A}_1 \ldots \mathbf{A}_{M-1}]$ are the M subband transform matrices which collectively describe the overall transform applied to the input, and $\mathbf{f}_M = [f_M(i)] = [\mathbf{g}_0{}^t\ \ \mathbf{g}_1{}^t \ldots \mathbf{g}_{M-1}{}^t\,]^t$ contains the N nonzero coefficients of all M subfilters, partitioned in proper order.

For example, consider the 2-branch decomposition of an 8-tap FIR filter as shown in Figure 1. Here N = 8 and M = 2, and therefore, the number of nonzero coefficients in each subfilter is P = N/M = 4. In this case,

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}, \tag{6}$$

$$\mathbf{f}_2 = \begin{bmatrix} \mathbf{g}_0^t & \mathbf{g}_1^t \end{bmatrix}^t = \begin{bmatrix} g_0(0) & g_0(1) & g_0(2) & g_0(3) & g_1(0) & g_1(1) & g_1(2) & g_1(3) \end{bmatrix}^t. \tag{7}$$

We refer to $\mathbf{T}_M$ as the decomposition matrix since it governs the effect of structural decomposition on the input data. The decomposition matrix and the vector $\mathbf{f}_M$ will be directly useful for our study of adaptive filters.

The decomposition matrix $\mathbf{T}_M$ has several unique and interesting properties. It is interesting to note that

$$\mathbf{T}_M = \underbrace{\mathbf{T}_2\mathbf{T}_2 \ldots \mathbf{T}_2}_{\text{L-times}} = (\mathbf{T}_2)^L, \tag{8}$$

where $M = 2^L$. In the limiting case when $N = M = 2^L$, $\mathbf{T}_N = \mathbf{R}_N$, and the WHT results as a special case of the class of transforms described in this paper. Thus, the decomposition matrix can be used to realize the WHT when possible, or generate structures with different number of subbands in the more general case. The decomposition matrix $\mathbf{T}_M$ is orthogonal since $(\mathbf{T}_M)^t\,\mathbf{T}_M = M\mathbf{I}_M$, where $\mathbf{I}_M$ is the M-dimensional identity matrix.

Another feature of significant importance is the efficient hardware realization of $\mathbf{T}_M$. The proposed structure requires no multiplications and the transform $\mathbf{T}_M$ can be directly and efficiently implemented using multiplier-less networks. For example, for the 8-tap filter case considered earlier, direct implementation of the 8×8 Hadamard transform requires 56 additions/ subtractions. However, 3 consecutive operations using $\mathbf{T}_2$ requires only 24 additions/ subtractions. In general, the N dimensional WHT requires $N(N-1) = N^2 - N$ additions/ subtractions, whereas, the fast algorithm implementing the same transform requires $N \log_2 N$ arithmetic operations. The hardware of the interpolator network can be modified to further reduce the number of arithmetic operations [6].

Assuming that the number of subbands is implicitly known, we drop the subscripts on the decomposition matrix and the subfilter coefficient vector, and denote them by T and f respectively for the sake of simplicity in future discussions. Furthermore, the time index (n) is introduced to denote the time-varying characteristics of the adaptive filter tap. With these modifications, the overall filter vector is given by $\mathbf{h}(n) = \mathbf{T}\,\mathbf{f}(n)$. The operation of the filter on the input data vector $\mathbf{x}(n)$ is then $\mathbf{h}^t(n)\mathbf{x}(n) = \mathbf{f}^t(n)\mathbf{T}^t\mathbf{x}(n) = \mathbf{f}^t(n)\mathbf{v}(n)$ where $\mathbf{v}(n) = \mathbf{T}^t\mathbf{x}(n)$ is the transformed data vector at time n. It should be noted that T can also represent any general transform suitable for subband decomposition.

ADAPTIVE FILTERS USING SUBBAND STRUCTURE

The proposed structure of the subband adaptive filter is shown in Figure 2, where the input transform T represents the subband decomposition. To derive the adaptive filtering algorithms, we consider the Wiener solution for the transformed structure. The mean-square error (MSE) for this structure is given by

$$\varepsilon \equiv E\left[e^2(n)\right], \tag{9}$$

where the error e(n) is

$$e(n) = d(n) - y(n), \tag{10}$$

and the output y(n) is

$$y(n) = \mathbf{v}^t(n)\,\mathbf{f}(n). \tag{11}$$

In above, d(n) is the desired signal and v(n) is the transformed input data. Thus,

$$\mathbf{v}(n) = \left[v_0(n) \quad v_1(n) \quad \ldots \quad v_{N-1}(n)\right]^t = \mathbf{T}^t\,\mathbf{x}(n). \tag{12}$$

As before, the subfilter coefficients are represented by

$$\mathbf{f}(n) = \left[\mathbf{g}_0^t(n) \quad \mathbf{g}_1^t(n) \quad \ldots \quad \mathbf{g}_{M-1}^t(n)\right]^t. \tag{13}$$

It should be noted that f contains the N coefficients of all M subfilters.

We can obtain the optimum Wiener solution by minimizing ε resulting in

$$\Phi_{vv}\mathbf{f}_{opt} = \mathbf{p}_{vd}, \quad \mathbf{f}_{opt} = \Phi_{vv}^{-1}\mathbf{p}_{vd}, \tag{14}$$

where

$$\Phi_{vv} = E\left[\mathbf{v}(n)\mathbf{v}^t(n)\right], \quad \mathbf{p}_{vd} = E[d(n)\mathbf{v}(n)]. \tag{15}$$

It is straightforward to verify that

$$\Phi_{vv} = \mathbf{T}^t\Phi_{xx}\mathbf{T}, \qquad \mathbf{p}_{vd} = \mathbf{T}^t\mathbf{p}_{xd}, \qquad \mathbf{f}_{opt} = \frac{1}{M}\mathbf{T}^t\mathbf{h}_{opt}, \tag{16}$$

where Φ_{xx} is the data correlation matrix, $\mathbf{p}_{xd}$ is the correlation between the data and the desired signal, and $\mathbf{h}_{opt}$ is the conventional optimum Wiener filter. Using this transformed Wiener solution, we can easily derive the adaptive algorithms for our subband structure.

The LMS algorithm for TDAF must have power normalization for improved convergence [4]. The "normalized LMS (NLMS)" [5] for our transformed structure is:

$$\mathbf{f}(n+1) = \mathbf{f}(n) + \gamma\hat{\Phi}_{vv}^{-1}(n)\mathbf{v}(n+1)e(n+1), \tag{17}$$

where $\hat{\Phi}_{vv}^{-1}(n)$ is a diagonal matrix defined as

$$\hat{\Phi}_{vv}^{-1}(n) = \mathrm{diag}\left[\hat{V}_0^{-1}(n) \quad \hat{V}_1^{-1}(n) \quad \ldots \quad \hat{V}_{N-1}^{-1}(n)\right], \tag{18}$$

and its diagonal elements

$$\hat{V}_j(n) = \beta\hat{V}_j(n-1) + (1-\beta)(v_j^2(n)), \qquad 0 < \beta < 1, \tag{19}$$

are estimates of the input power at j-th tap. The corresponding NLMS algorithm for the conventional filter structure is given by

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \frac{\alpha}{\hat{\sigma}_x^2(n)}\mathbf{x}(n+1)e(n+1), \tag{20}$$

where $\hat{\sigma}_x^2(n)$ is the estimates of the input signal power given by

$$\hat{\sigma}_x^2(n) = \beta\hat{\sigma}_x^2(n-1) + (1-\beta)(x^2(n)), \qquad 0 < \beta < 1. \tag{21}$$

The choice of parameters α, β, and γ as well as the initial conditions for $\hat{\sigma}_x^2(0)$ and $\hat{V}_j(0)$ are important to ensure good performance.

The effective convergence factor $\alpha / \hat{\sigma}_x^2$ and $\gamma / \hat{V}_j$ are time-varying and stabilize operation of the transform domain LMS. If the power in a specific band is too small the step size becomes too large resulting in adaptation failure. To prevent such situations, an adaptation disabling mechanism [4] for a specific band is possible.

The LMS adaptive filters using our decomposition structure have the following distinctive features compared to the existing WHT-based TDAF:

- The proposed method can be applied to any filter with even number of taps, whereas, the conventional WHT-based TDAF can be applied only to filters with $N = 2^L$ taps.
- The number of bands may vary by choice between 1 and 2^L for $N = 2^L P$ tap filter. However,

WHT-based TDAF requires a full $N = 2^L$ band to be used. Thus, the proposed scheme allows more flexibility.

- Finally, the decomposition technique utilizes simple hardware to realize the input transform T. The hardware realization of fast WHT occurs as a special case.

BEHAVIOR OF SUBBAND ADAPTIVE FILTERS

We now consider application of the subband adaptive filters for the system identification problems, and compare the results to that of the conventional adaptive filters.

Problem Description and Simulation Methodology

The performance of various adaptive algorithms and structures can be compared using standardized simulation methods. One popular strategy for the adaptive filters is to test their convergence behavior on the system identification problem, which adequately quantifies the performance of most algorithms. Our simulation methods are adapted from Mulgrow and Cowan [7]. The system identification problem shown in Figure 3 is tackled using various decomposition structures and components such as input data and additive noise.

It should be noted that one way of specifying the desired signal d(n) in simulation is to equate it with the impulse response of the unknown system $\hat{\mathbf{h}}$. In our work, $\hat{\mathbf{h}}$ is set to an arbitrary 16×1 unit norm vector whose frequency response is shown in Figure 4. It is also important to choose a suitable performance measure for comparing various algorithms and structures. One widely used measure is the squared error $e^2(n)$. However, this quantity is signal dependent and thus does not solely represent the filter performance. A more practical measure is the statistical norm:

$$\rho(n) = E\left[\left(\mathbf{h}(n) - \hat{\mathbf{h}}(n)\right)^t \left(\mathbf{h}(n) - \hat{\mathbf{h}}(n)\right)\right]. \tag{22}$$

In practice, the expectation is estimated by the ensemble average over several simulation trials (20 runs were used in our work).

The characteristics of the input signal are equally important. For a given filter, convergence rate varies for different input signals. The most significant factor is the degree of correlation in the input data. As mentioned in the previous sections, the goal of input transform in adaptive filtering is to reduce the correlation in the data. To investigate the effect of transforms on the convergence rate, it is necessary to control the amount of color in the signal. Towards this end, in Figure 3, zero-mean unit-variance white Gaussian signal $\zeta(n)$ is filtered by a 3-tap FIR "coloring channel". Thus the filtered signal x(n) has varying degree of correlation determined by the channel. The coefficient vectors $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ for each channel are:

$$\mathbf{c}_1 = [1 \quad 0 \quad 0]^t, \quad \mathbf{c}_2 = [0.2602 \quad 0.9289 \quad 0.2602]^t, \quad \mathbf{c}_3 = [0.3842 \quad 0.8704 \quad 0.3842]^t. \tag{23}$$

The eigenvalue ratio (EVR) of the autocorrelation matrix $\mathbf{R}_{xx}$ of the input data corresponding to these channels are 1.0, 11.8, and 68.6 [7]. The degree of correlation in the data increases with EVR. The coefficients of the three channels and the unknown system are normalized to

$$\sum_{i=0}^{n} c_i^{\,2} = 1, \tag{24}$$

so that the variance $\sigma_x^{\,2}$ of the data $\mathbf{x}(n)$ is same as $\sigma_\zeta^{\,2}$ and equals 1. This choice of $\sigma_x^{\,2}$ is convenient to determine the initial input power estimate $\hat{\sigma}_x^{\,2}(0)$ required for the NLMS algorithm. The dimension of $\mathbf{R}_{xx}$ is determined by the number of taps in the adaptive filter. In our simulation, tap number is set equal to 16.

The power of the zero mean white Gaussian driving noise source $\zeta(n)$ may affect the convergence behavior. To eliminate this effect we set the input data x(n) to have a unity variance and the unknown system to have a unity gain. The output "estimation error" therefore converges to the power (variance) of the additive noise $v(n)$ which serves as a "floor of convergence". We used different noise floors set at -70 dB, -90 dB, and -30 dB, respectively.

Simulation Results

As we now show, the parameters affecting the convergence rate are the decomposition structure (number of subbands) of the adaptive filter, the input correlation (expressed by EVR), and the additive noise power. The convergence behavior of the adaptive filter for system identification is now studied as a function of these parameters.

Performance of the Subband LMS Filter

We first use the normalized LMS (NLMS) algorithm to compare the performance of the conventional (i.e. 1-band) 16 tap, 4-band, and full 16-band adaptive filters. Lee and Un [5] described a method to choose parameters associated with the NLMS algorithm. Following their guidelines, the parameters used in our NLMS simulation are:

- The convergence factor $\alpha = \gamma = 1/N$ for all cases ($N = 16$ is the total number of taps in our example).

- The factor β is chosen as $1-\gamma$.

- The initial value of variance estimation for the single band case, $\hat{\sigma}_x^2(0)$, must be larger than $\delta_0 = \gamma N \sigma_x^2$. In our simulation, the variance σ_x^2 of the input signal is set to 1, as mentioned previously. Therefore $\delta_0 = \frac{1}{N} \cdot N \cdot 1 = 1$, and $\hat{\sigma}_x^2(0)$ must be equal to or larger than 1. The corresponding initial variance for the M-band case is M times larger than the 1-band case, i.e.

$$\hat{V}_j(0) = M \cdot \hat{\sigma}_x^2(0) = M, \qquad 0 \le j \le N-1. \tag{25}$$

The reason for this multiplication is that our transform matrix T is not normalized ($T\,T^t = MI$) and the power estimate for each band is on the average M times larger than the 1-band case. If the initial value of the variance estimate is too small, the convergence step (which is inversely proportional to the variance) will be too large, and an overshoot is likely to occur. This overshoot results in a slower convergence [5].

The simulations of the NLMS algorithm for different input channels and additive noise are shown in Figures 5 through 9. Figure 5 represents the results with noise channel 1 (no color) and -70 dB power. Clearly, the 16 band structure converges the fastest. The 4-band case also demonstrates better performance than the conventional 1-band filter. However, since the data is essentially white with a small EVR (≈ 1), the convergence rates for the subband structure and the conventional approach are similar.

Figure 6 shows the result with noise channel 2 (moderately colored case) with a -70 dB noise power. Here the effect of introducing some color is evident. Once again, the 16-band decomposition performs the best. Note that the extent of correlation in the input is moderate corresponding to an EVR of 11.8. The 16-band structure succeeds in whitening the input data and converges the fastest (in approximately 500 iterations), while the conventional 1-band filter is relatively slow and converges slowest (in approximately 650 iterations).

The results are more dramatic in Figure 7 for the highly colored input data produced by channel 3 (EVR = 68.6). In this case, the conventional 1-band filter converges very slowly (about 3000 iterations), while the 4-band and 16-band filters converge much faster (within 900 iterations). The 16-band filter performs the fastest as in the prior experiments since the full-band decomposition matrix T_{16} decorrelates the data best. However, it should be noted that the convergence rate for the 16-band case is still slower than the case when the input data is white. The reason is that the Hadamard transform is successful in partially whitening the input but is not optimal for this purpose like the Karhunen-Loeve

Transform (KLT) [2]. Nonetheless, since the KLT is impractical for on-line filtering, our subband structure is very attractive for practical applications. We have shown that it dramatically improves the performance of the NLMS algorithm when the input data is colored.

Figure 8 demonstrates the effect of lowering the additive noise level to -90 dB. Interestingly, there is a greater difference between the 16-band and 4-band convergence rates for a smaller noise level. The conventional 1-band NLMS algorithm converges almost at the same rate as for its -70 dB case. The 1-band filter achieves the "noise floor" well after 1000 iterations. On the other hand, the 16-band filter performs significantly better than even the 4-band case. The reason again is the successful decorrelation of the input by the full subband structure. Although not shown in Figure 8, all three filters achieve the minimum $\rho(n)$ of about -90 dB, with the 16-band case converging significantly ahead of others.

Figure 9 shows the convergence rates when the additive noise level is increased to -30 dB, and the colored input data generated by channel 3 is used. For higher additive noise power, the subband structure still converges much faster than the conventional 1-band case. However, for uncorrelated data generated by channel 1, no significant difference was observed between the performance of the conventional 1-band filter and the subband structure. In general, it is advantageous to use the subband structure since any correlation in the input data is handled by the multiband filter.

CONCLUDING REMARKS

We have shown that the structural subband decomposition of FIR filters is highly suitable for adaptive filtering applications. The relation between the structural subband decomposition and the input transform was established and the corresponding transform matrix was shown to be orthogonal with elements which are either 1 or -1. Using our theory, an adaptive algorithm was developed for the subband structure. The simulation results have demonstrated the improvements in the convergence rates for the system identification problem.

The most dramatic effect of the structural subband decomposition has been shown in the case of NLMS algorithm with highly colored input data. It is interesting to note that a relatively small number of bands significantly improves the convergence speed. Our method allows the number of bands to be varied as necessary, including the 1-band conventional filter and the fullband WHT-based TDAF as a special case. This is particularly useful when the filter length is not a power of 2 and the WHT or the fullband decomposition can not be used. The proposed method essentially generalizes the WHT-based TDAF algorithm.

Acknowledgements

This work was supported in part by a University of California MICRO grant with matching supports from the Alcatel-Telettra Corporation and the Martin-Marietta Corporation.

References

[1] S. S. Narayan, A. M. Peterson, and M. J. Narashima, "Transform domain LMS algorithm," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 609-615, June 1983.

[2] W. K. Jenkins and D. F. Marshall, "On the analysis and design of unitary transformations for adaptive FIR filters," *Proc. 29th Midwest Symp. on Circuits & Systems*, Lincoln, NE, pp. 534-537, August 1986.

[3] W. K. Jenkins and J. R. Kreidle, "Adaptive digital filters using the Walsh-Hadamard transform," *Proc. 1986 IEEE International Symp. on Circuits & Systems*, San Jose, CA, pp. 875-878, May 1986.

[4] D. F. Marshall, W. K. Jenkins, and J. J. Murphy, "The use of orthogonal transform for improving the performance of adaptive filters," *IEEE Trans. on Circuits & Systems*, vol. CAS-36, pp. 474-478, April 1989.

[5] J. C. Lee and C. K. Un, "Performance of transform-domain adaptive filters," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. ASSP-34, pp. 499-510, June 1986.

[6] S. K. Mitra and A. Mahalanobis, "Efficient FIR filter design and implementation using structural subband decomposition," *Proc. 1990 Bilkent International Conference on New Trends in Communications, Control, and Signal Processing,* Ankara, Turkey, pp. 1005-1022, July 1990.

[7] B. Mulgrow and C. F. Cowan, *Adaptive Filters and Equalizers,* Kluwer Academic Publishers, Norwell, MA, 1988.

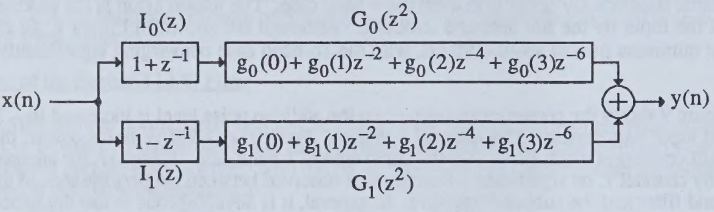

$I_0(z)$
$G_0(z^2)$
$1+z^{-1}$
$g_0(0)+g_0(1)z^{-2}+g_0(2)z^{-4}+g_0(3)z^{-6}$
x(n)
y(n)
$1-z^{-1}$
$g_1(0)+g_1(1)z^{-2}+g_1(2)z^{-4}+g_1(3)z^{-6}$
$I_1(z)$
$G_1(z^2)$

Figure 1 - Two-band decomposition of an 8-tap FIR filter.

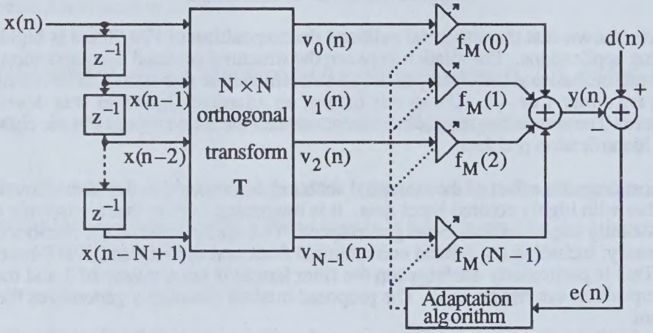

x(n)
z^{-1}
x(n−1)
z^{-1}
x(n−2)
z^{-1}
x(n−N+1)
N×N
orthogonal
transform
T
$v_0(n)$
$v_1(n)$
$v_2(n)$
$v_{N-1}(n)$
$f_M(0)$
$f_M(1)$
$f_M(2)$
$f_M(N-1)$
d(n)
y(n)
+
−
e(n)
Adaptation
algorithm

Figure 2 - Proposed adaptive filter structure.

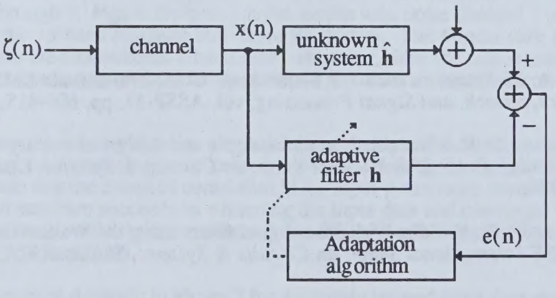

noise v(n)
ζ(n)
channel
x(n)
unknown
system $\hat{h}$
+
−
adaptive
filter h
e(n)
Adaptation
algorithm

Figure 3 - Simulation structure.

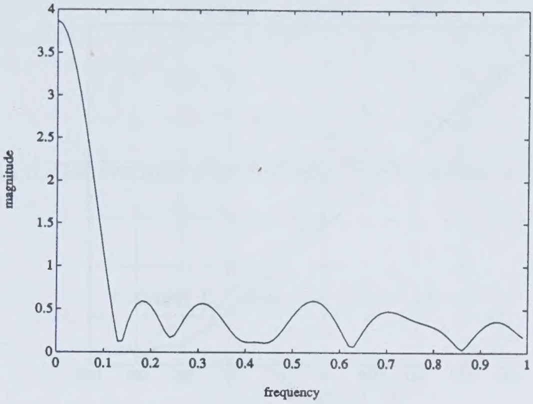


Figure 4 - Frequency response of unknown system.

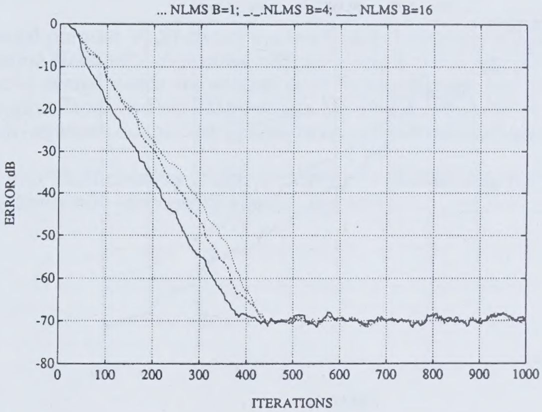


Figure 5 - NLMS result (channel 1, noise = -70 dB).

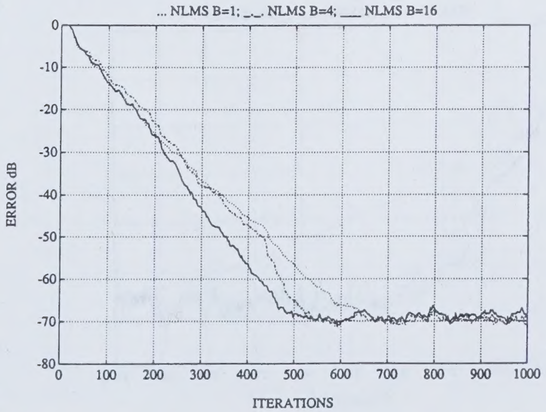


Figure 6 - NLMS result (channel 2, noise = -70 dB).

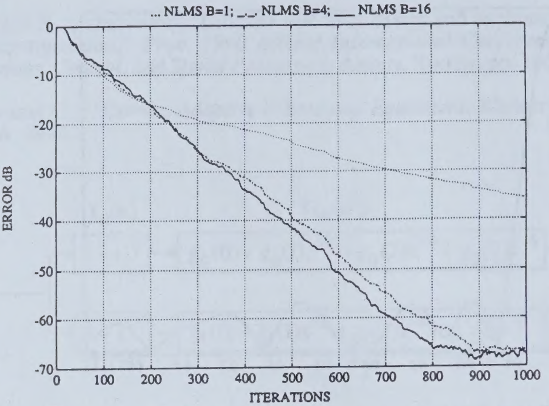


Figure 7 - NLMS result (channel 3, noise = -70 dB).

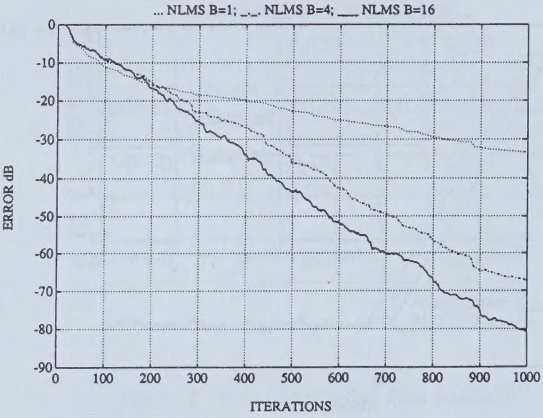


Figure 8 - NLMS result (channel 3, noise = -90 dB).

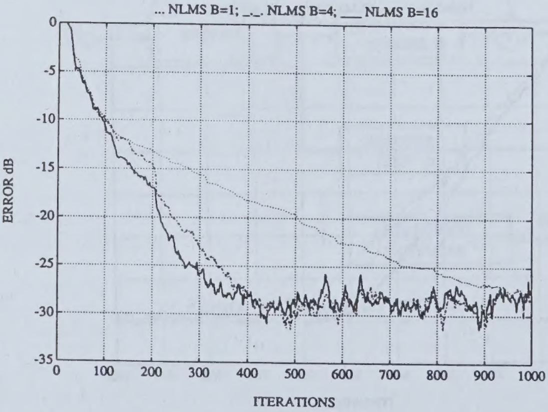


Figure 9 - NLMS result (channel 3, noise = -30 dB).

ADAPTIVE FILTER DESIGN FOR A HIGH PERFORMANCE ECHO CANCELLER

Richard C.Younce and Charles E. Rohrs

Tellabs Research Center
3702 N. Main St. Bld 2
Mishawaka, IN 46545
(219)-259-9818

Abstract A special purpose VLSI circuit has been designed to cancel echos in the PSTN (Public Switched Telephone Network). A normalized gain, leaky Least Mean Squares (LMS) algorithm is employed. Several design issues are address including windowed verses non-windowed LMS, finite register length effects, and the adaptive gain performance effects. A fault detection algorithm is unveiled which improves double talk performance while maintaining good convergence rates.

Implementation into VLSI circuitry is also discussed. A flexible chip designed allowing a shorter design cycle and lower non-reoccurring cost is presented.

ON THE TRACKING CAPABILITIES OF OPTIMAL BOUNDING ELLIPSOID ALGORITHMS[1]

Y.F. Huang[†] *and J.R. Deller, Jr.*[‡]

[†]University of Notre Dame; Department of Electrical Engineering; Notre Dame, IN 46556 huang@saturn.ece.nd.edu

[‡]Michigan State University; Department of Electrical Engineering / 260EB; East Lansing, MI 48824-1226 deller@ee.msu.edu

Abstract: A unified family of OBE algorithms for recursive parameter estimation is introduced and general analysis of the tracking properties of this class of methods is presented. Rescue schemes are proposed to effectively compensate for practical violations of the theory caused by time-variance. Two representative OBE methods are used to illustrate the tracking characteristics of such algorithms through simulation.

1 Introduction

Adaptive filtering techniques are often implemented in practice to track time-varying system parameters. Much attention has been paid to this problem in the adaptive filtering literature, with analysis of varying degrees of rigor being performed mainly for the LMS and RLS algorithms (see, e.g., [1,2,3,4,5]).

This paper investigates tracking properties of a general class of recursive estimation methods known as the *optimal bounding ellipsoid (OBE)* algorithms [6,7]. The OBE algorithms belong to a broader class of estimation algorithms termed *bounded error* methods and similar names, which, in turn, are part of a larger field known as *set-membership (SM) parameter estimation* and similar names (see, e.g., [8,9,10]). Of the myriad of SM approaches, the OBE algorithms are most readily applicable to problems in real-time signal processing and control due to their relatively low computational complexity and their similarity to widely-used identification methods. In fact, at first glance, all OBE algorithms appear to be very similar to the (weighted) recursive least squares (RLS) algorithm. However, in contrast to the RLS algorithm which obtains an optimal (in the sense of least squares error (LSE)) point estimate for the underlying problem, the OBE algorithms are developed using a set-theoretic framework. This causes the algorithm to behave quite differently from the RLS algorithm in many ways. In addition, the algorithm incorporates a novel data-dependent weighting approach which results in selective updating.

Like all SM methods, the OBE algorithms seek a *membership set* in the parameter space, say E_t at time t, which is guaranteed to contain all solutions which are compatible with the model of the underlying process, the assumptions on noise, and the observation data. If the set of *only* those solutions which are compatible is denoted by $\bar{E}_t$, then it is the case that $\bar{E}_t \subseteq E_t$. (Some of the more computationally complex SM methods track $\bar{E}_t$ exactly.) In the OBE cases, the membership set E_t takes the form of a bounding hyperellipsoid. However, the fundamental development of the OBE algorithms assures

---

[1] **Acknowledgments:** JD was supported by the National Science Foundation under Grant No. MIP-9016734 and by the Office of Naval Research under Grant No. N00014-91-J-1329, and also by an Ameritech Fellowship. The authors wish to thank Dr. A.K. Rao at COMSAT for valuable discussions and Mr. Wenjun Zeng at University of Notre Dame for providing computer simulation programs.

that the feasible set of solutions contains the true system parameters only if the system is time invariant. In the case of time-varying systems, it is important to ensure that the time-varying true parameters, say $\{\theta_o(t)\}$, are contained in the bounding ellipsoids $\{E_t\}$. In this paper, such conditions will be presented. It will also be shown that if a jump in the true parameter vector $\theta_o(t)$ causes it to fall outside the bounding ellipsoid, then, provided that the jump is not too large, the bounding ellipsoids will move toward $\theta_o(t)$ and eventually enclose $\theta_o(t)$ again. Rescue schemes are proposed which will guarantee the existence of bounding ellipsoids in the event of large parameter variations.

Simulation results are presented to show that two representative OBE algorithms are able to track slow and abrupt variations in the parameters. The tracking performance in terms of parameter estimation error is comparable to the RLS algorithm with a forgetting factor. Abrupt changes in the parameter can in some cases be tracked better by the OBE algorithms than by the RLS algorithm. Further results are found in [9,11,12,13].

2 A Family of OBE Algorithms

The seminal paper in the development of the OBE methods is that of Fogel and Huang [6]. We shall refer to the original algorithm as F-H/OBE. The other example algorithm which will also be used as a basis for discussion in this paper is due to Dasgupta and Huang (D-H/OBE) [7]. Other examples are cited in Table 1. We shall first place all OBE methods into a unifying framework which has been called the *unified OBE (UOBE)* algorithm [14]. We will relate each of the above to the general framework, then focus on F-H/OBE and D-H/OBE as representative algorithms for the study of the main issue — tracking capability.

The model structure considered here is the following ARX model:

$$y(t) = \theta_o^T \phi(t) + v(t) \tag{2.1}$$

where $\theta_o = [a_1 a_2 \dots a_n b_0 b_1 \dots b_m]^T$ is the true parameter vector and

$$\phi(t) = [y(t-1) y(t-2) \dots y(t-n) u(t) u(t-1) \dots u(t-m)]^T$$

is the measurable regressor vector. The noise $v(t)$ is assumed to be uniformly bounded in magnitude with a known bound γ, i.e.,

$$|v(t)| \leq \gamma \tag{2.2}$$

Assume that at time instant $t-1$, the exact membership set $\bar{E}_{t-1}$ is outer-bounded by the ellipsoid E_{t-1} described by

$$E_{t-1} = \left\{\theta \in R^N : [\theta - \theta(t-1)]^T P^{-1}(t-1)[\theta - \theta(t-1)] \leq \sigma^2(t-1)\right\} \tag{2.3}$$

where $N = n+m+1$, $P^{-1}(t-1)$ is a positive definite matrix, and $\theta(t-1)$ is the center of the ellipsoid. At time instant t, the observation $y(t)$ yields a set S_t which is a degenerate ellipsoid in the parameter space, namely

$$S_t = \{\theta \in R^N : [y(t) - \theta^T \phi(t)]^2 \leq \gamma^2\} \tag{2.4}$$

From (2.1) and (2.2) it is clear that S_t contains the true parameter vector. An ellipsoid E_t which contains the intersection of E_{t-1} and S_t is then given by (see [6] and [9])

$$\begin{aligned} E_t = \Big\{\theta \in R^N : \quad & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (2.5) \\ \alpha_t[\theta - \theta(t-1)]^T P^{-1}(t-1)[\theta - \theta(t-1)] + \beta_t[y(t) - \theta^T \phi(t)]^2 &\leq \alpha_t \sigma^2(t-1) + \beta_t \gamma^2 \Big\} \end{aligned}$$

where α_t and β_t are positive numbers (sequences on t) to be specified below. By performing some algebraic manipulations on (2.5) and employing the matrix inversion lemma, a recursive set of equations for updating the ellipsoid is found to be

$$E_t = \{\theta \in R^N : [\theta - \theta(t)]^T P^{-1}(t)[\theta - \theta(t)] \leq \sigma^2(t)\} \tag{2.6}$$

where

$$P(t) = P_\alpha(t-1) - \beta_t \frac{P_\alpha(t-1)\phi(t)\phi^T(t)P_\alpha(t-1)}{1 + (\beta_t/\alpha_t)G(t)}; \quad P_\alpha(t-1) \overset{\text{def}}{=} \frac{P(t-1)}{\alpha_t} \tag{2.7}$$

$$\sigma^2(t) = \alpha_t\sigma^2(t-1) + \beta_t\gamma^2 \frac{\beta_t\delta^2(t)}{1 + (\beta_t/\alpha_t)G(t)} \tag{2.8}$$

$$\theta(t) = \theta(t-1) + \beta_t P(t)\phi(t)\delta(t) \tag{2.9}$$

in which, for convenience, we have defined $\delta(t) \overset{\text{def}}{=} y(t) - \phi^T(t)\theta(t-1)$ (the residual sequence), and $G(t) \overset{\text{def}}{=} \phi^T(t)P(t-1)\phi(t)$. Equations (2.6)—(2.9) characterize the update of the bounding ellipsoids. The center $\theta(t)$ of the bounding ellipsoid E_t can be taken to be a point estimate of the parameter vector. In fact, it is not difficult to see that $\theta(t)$ is the LSE estimate of the parameters at time t, given a sequence of forgetting factors $\{\alpha_t\}$ and data weights $\{\beta_t\}$ (see, e.g., [15]).

Different sequences $\{\alpha_t\}$ and $\{\beta_t\}$ yield different sequences of parameter estimates and ellipsoids. To ensure meaningful behavior of the algorithm, the sequences $\{\alpha_t\}$ and $\{\beta_t\}$ must be chosen to optimize the bounding ellipsoids in some sense. It is the combination of weighting sequences and optimization strategy which distinguishes the various OBE algorithms. Formally, an OBE algorithm consists of the set of recursions (2.7)–(2.9) (with α_t and β_t replaced by optimal values, say α_t^* and β_t^*) in conjunction with an optimization criterion for choosing these weights. The optimization should intuitively be such that the "size" of E_t is diminished in some sense at each t. Three optimization criteria have been used. These are [6,7]: (1) Minimize the the determinant of the inverse ellipsoid matrix, $\mu_{vol}(t) \overset{\text{def}}{=} \det\{\sigma^2(t)P(t)\}$; (2) Minimize the trace of the inverse ellipsoid matrix, $\mu_{tr}(t) \overset{\text{def}}{=} \text{trace}\{\sigma^2(t)P(t)\}$; and, (3) Minimize the parameter $\sigma^2(t)$. Note that $\mu_{vol}(t)$ is proportional to the square of the volume of the ellipsoid, while $\mu_{tr}(t)$ is proportional to the sum of squares of its semi-axes. Criterion 3, while controversial in terms of its interpretability (see [14,15]), has been used in conjunction with a specific weighting strategy in D-H/OBE to achieve a rigorous proof of convergence in a certain sense. To date, D-H/OBE is the only published OBE algorithm with a theoretical proof of convergence of the bounding set.

Having established the optimization criteria, let us now focus on the weight sequences. For any of the criteria above, there is only one quantity to be optimized at time t which in each case is dependent upon both α_t and β_t. However, the numbers α_t and β_t are essentially independent of one another, so that any attempt to optimize one of the criterion measures with respect to *both* α_t and β_t results in an infinity of solutions which is resolved by arbitrarily choosing a value of either weight. Accordingly, we may either tie the weights together through some functional relation, optimize over only one weight and choose the other according to some predetermined purpose, or simply eliminate the "unused" weight altogether by setting it to unity (a special case of the second strategy). We shall adopt the policy of writing the weights α_t and β_t as functions of a single parameter to be optimized at time t, say λ_t, so that (in conventionally abusive notation) $\alpha_t = \alpha_t(\lambda_t)$ and $\beta_t = \beta_t(\lambda_t)$, where $\{\alpha_t\}$ and $\{\beta_t\}$ should now be considered to be sequences of *functions*. The example OBE algorithms to be cited below will provide examples of such function sequences. These function sequences are not arbitrary, but must be such that their ratio, say $q_t(\lambda_t) = \beta_t(\lambda_t)/\alpha_t(\lambda_t)$, is a continuous, one-to-one mapping $q_t : (0, d_t) \longrightarrow (0, \infty)$ with $d_t > 0$. It is also assumed that both $\alpha_t(\lambda_t)$ and $\beta_t(\lambda_t)$ are positive on $(0, d_t)$, and that neither is defined outside of this domain. For sufficient generality, it is important to note that these functions need *not* depend on λ_t. For example, $\{\alpha_t\}$ may be chosen *independently* of the optimization in which case, at time t, $\alpha_t(\lambda_t) = \alpha_t$, a constant (function) independent of λ_t. It is natural to refer to λ_t as a "weight" at time t, since if $\alpha_t(\lambda_t) = 1$ and $\beta_t(\lambda_t) = \lambda_t$, then λ_t is simply the weight associated with the standard weighted RLS recursion with no forgetting factor. This general setup will allow us to embrace all OBE algorithms in a single theoretical framework.

Before enumerating specific OBE algorithms, we briefly describe a number of results which generalize and unify all optimization procedures found in the literature. The first concerns the process of finding an optimal value of λ_t, say λ_t^*, for a specified criterion. It is shown in [9,14] that, if it exists, λ_t^* is the unique positive root in λ_t on the interval $(0, d_t)$ of equations $F_{vol}(q_t(\lambda)) = 0$ and $F_{tr}(q_t(\lambda)) = 0$ for the volume and trace measures respectively, where $F_{vol}(s)$ is quadratic, and $F_{tr}(s)$ is cubic, in s. The coefficients of these polynomials are given in terms of quantities which are known prior to time t. A much less compact and elegant generalization arises for σ^2 minimization. However, for the one notable example of σ^2 minimization, D-H/OBE, a simple quadratic in λ results which must be solved for a positive root on $(0, d]$ with $0 < d < 1$. Whichever criterion is used, the optimal weights α_t^* and β_t^* to be used in the recursions (2.7)–(2.9) are taken to be

$$\alpha_t^* = \begin{cases} \alpha_t(\lambda_t^*), & \lambda_t^* \in (0, d_t) \text{ exists} \\ 1, & \text{otherwise} \end{cases} \quad \text{and} \quad \beta_t^* = \begin{cases} \beta_t(\lambda_t^*), & \lambda_t^* \in (0, d_t) \text{ exists} \\ 0, & \text{otherwise} \end{cases}. \tag{2.10}$$

Fast methods which do not require root finding are available to check for the existence of an optimal solution (see [9] or papers describing specific algorithms cited in Table 1).

A second set of unifying principles appears in [9,14]. These papers show that volume and trace algorithms are theoretically independent of the choices of sequences $\{\alpha_t(\lambda_t)\}$ and $\{\beta_t(\lambda_t)\}$ in certain key ways. In particular, the following result is proven:

Theorem 1 *For volume or trace minimization, none of the following important quantities depends on the choices of sequences* $\{\alpha_t(\lambda_t)\}$ *and* $\{\beta_t(\lambda_t)\}$*: (1) the resulting sequence of measures of optimality (*$\{\mu_{vol}(t)\}$ *or* $\{\mu_{tr}(t)\}$*); (2) the data points selected (times for which there exists* $\lambda_t^* > 0$*); and, (3) the sequence of parameter estimates,* $\{\theta(t)\}$*. However, in an OBE algorithm with* σ^2 *minimization, all of these items depend on the sequences* $\{\alpha_t(\lambda_t)\}$ *and* $\{\beta_t(\lambda_t)\}$*.*

We are now prepared to enumerate the various OBE algorithms that have been reported in the literature. These are shown in Table 1. Specific optimization procedures can be found in [9,14] or by reference to the original papers for each algorithm (citations in Table 1). Likewise we shall leave the discussion of relative advantages of the various methods and other practical details to the reader's further pursuit of the literature.

3 Analysis of Tracking Characteristics

Tracking in the context of OBE algorithms means ensuring that the time varying true parameter vector is contained in the bounding ellipsoid. The theorems below present conditions under which parameter tracking can be accomplished. These theorems and their proofs are direct generalizations of those presented in [13], and the proofs are omitted here.

Lemma 1 *A sufficient condition for $\theta_o(t) \in E_t$ is*

$$(\theta_o(t) - \theta(t-1))^T P^{-1}(t-1)(\theta_o(t) - \theta(t-1)) \le \sigma^2(t-1) \tag{3.1}$$

Theorem 2 *At any time instant t, the true parameter $\theta_o(t) \in E_t$ if and only if*

$$(\theta_o(t) - \theta(t-1))^T P^{-1}(t-1)(\theta_o(t) - \theta(t-1)) \le \sigma^2(t-1) + \frac{\beta_t^*}{\alpha_t^*}(\gamma^2 - v^2(t)) \tag{3.2}$$

where $v(t)$ is the noise term in (2.1).

It is easy to see from Lemma 1 that if the true parameter $\theta_o(t)$ is constant for all t, then the bounding ellipsoids obtained by the OBE algorithms enclose $\theta_o(t)$ at all time instants. This is a property that all well-devised OBE algorithms should have when applied to estimation of time-invariant parameters. If, on the other hand, $\theta_o(t)$ is time-varying, and if at some time instant t, $\theta_o(t)$ is found to be out of the bounding ellipsoid E_t, it must not have been included in E_{t-1}. Theorem 2 then demarcates the region in which $\theta_o(t)$ can migrate without loss of tracking. This theorem also shows that by choosing γ^2 to be larger than the actual bound, say $\bar{\gamma}^2$, on $v^2(t)$, it is possible to increase the tracking capability of the algorithm. The next theorem gives an upper bound on the maximum variation in the parameters for which tracking is guaranteed.

Theorem 3 *If $\theta_o(t-1) \in E_{t-1}$, and $\beta_t^* \neq 0$, then $\theta_o(t) \in E_t$ if*

$$\| \Delta(t) \| \le e_{\min}^{-1}(t-1) \left\{ \sqrt{\frac{\beta_t^* e_{\min}(t-1)}{\alpha_t^* e_{\max}(t-1)}[\gamma^2 - \bar{\gamma}^2] + \sigma^2(t-1)} - \sqrt{\sigma^2(t-1)} \right\} \tag{3.3}$$

where $\Delta(t) \overset{\text{def}}{=} \theta_o(t) - \theta_o(t-1)$; $e_{\min}(t-1)$ and $e_{\max}(t-1)$ denote, respectively, minimum and maximum eigenvalues of $P^{-1}(t-1)$; and $\| \cdot \|$ denotes the Euclidean norm. The quantity $\bar{\gamma}^2$ is the actual bound on $v^2(t)$ and γ^2 is the chosen bound such that $\gamma^2 \ge \bar{\gamma}^2$.

It can be seen from (3.3) that if $\beta_t^* = 0$, then the difference between γ^2 and $\bar{\gamma}^2$ cannot be exploited to improve the tracking capability of the algorithm. In this case, $\theta_o(t) \in E_t$ if and only if $\theta_o(t) \in E_{t-1}$. Thus if $\theta_o(t)$ jumps out of E_{t-1}, and no updates are performed at future time instants $t+i$, then $\theta_o(t+i) \notin E_{t+i} = E_{t-1}$, and the shifting parameters may not be tracked. However, experimentally it is observed that an update will almost always occur in a finite interval of time. Further, heuristic quantitative analysis of one important OBE algorithm, D-H/OBE, found in [13] supports the occurrence of finite time updating. It is shown [13] that if $\theta_o(t)$ remains fixed after it jumps out of E_t, and if the jump is not large enough to cause the subsequent ellipsoids E_{t+i} to vanish, for $i \ge 0$, then the D-H/OBE algorithm guarantees that the true parameter will be tracked (enclosed) in finite time.

4 Rescue Schemes

In many cases when the parameter jump is large, or if the ellipsoid has shrunk to a very small size, the intersection of E_{t-1} and S_t can be void. In such cases $\sigma^2(t)$ will become negative, indicating that a bounding ellipsoid could not be constructed. To circumvent such a failure of the algorithm, some rescue schemes have been proposed. The first involves the increasing of the parameter $\sigma^2(t-1)$ thereby increasing the size of E_{t-1} so that the intersection of S_t and this enlarged E_{t-1} is no longer empty [13]. Accordingly, a valid ellipsoid E_t can then be constructed[2]. Further, for D-H/OBE, Rao and Huang [13] have specified precise changes needed to render $\sigma^2(t-1)$ positive in this scheme. The next section also presents a specific remedy for F-H/OBE.

The second procedure is a specific case of what Deller and Odeh [11] have called *selective forgetting.* This strategy partially or completely removes previous data from the present estimate according to certain user-defined criteria in order to remove the data's influence on the result. In the present context, the selection procedure monitors the parameter $\sigma^2(t-1)$ for positivity. When it is found that $\sigma^2(t-1) < 0$ for some t, previously accepted data sets are "back rotated," beginning with the oldest data remaining in the estimate, until this number becomes positive[3].

In either case, the rescue scheme is similar to the covariance resetting technique used in RLS algorithms to cope with time-varying systems. However, in the RLS case, a change in the parameters has to be detected by some other means before the covariance matrix can be reset whereas for the OBE algorithms, $\sigma^2(t)$ becoming negative is an automatic indicator of a change.

We next explore the tracking behavior of two representative OBE algorithms through simulation studies. The original F-H/OBE algorithm (with volume minimization) and the more recent D-H/OBE algorithm are chosen for illustration. According to Theorem 1, the performance of F-H/OBE should be a good indicator of the behavior of any OBE algorithm with volume minimization. D-H/OBE is also illustrated since it represents the alternative σ^2 minimization strategy, and since it is the OBE method whose convergence and tracking properties are most well-understood in analytical terms [7,12,13].

5 Simulation Examples

The model used is an ARX(2,1) model

$$y(t) = a_1 y(t-1) + a_2 y(t-2) + bu(t) + v(t)$$

The nominal values for the parameters are $a_1 = -0.4$, $a_2 = -0.85$, and $b = 0.5$. The noise sequence $\{v(t)\}$ and the input sequence $\{u(t)\}$ were both generated by a pseudo-random number generator with a uniform distribution in [-1,1]. For the D-H/OBE algorithm, we chose $d = 0.2$ [see discussion above (2.10)], $\gamma^2 = 1.0$, and $\sigma^2(0) = 100$. For the F-H/OBE algorithm, we chose $\gamma^2 = 1.0$ and $P(0) = 10^{-5}I$. In all the examples shown here, the parameter estimates are taken to be the centers of the optimal bounding ellipsoids. The parameters were varied as follows:

---

[2] Alternatively, γ^2 could be increased to permit a non-null intersection. However, the former procedure is preferable because it causes $\theta(t)$ to migrate towards $\theta_o(t)$, thereby reducing the parameter estimation error.

[3] The term "back rotate" arises in the context of an alternate form of the OBE recursions which is particularly amenable to this scheme. See, e.g., [9,11] for details.

Case 1: Slow Variation in the Parameter Vector. The parameters a_1, a_2, and b were varied by 1% every 10 samples, starting from the first sample, and the output data $\{y(t)\}$ were generated for $t = 1, 2, \ldots 1000$. It was then observed that the bounding ellipsoids created by both the F-H/OBE algorithm and the D-H/OBE algorithm contain the true parameter at all time instants. The parameter estimates, i.e., the centers of the ellipsoid, are plotted against the true parameters in Figs. 1(a) and 1(b).

Case 2: Jump in Parameter at $t = 400$. The parameter a_1 was changed by 50% at the 400^{th} sample, and b and a_2 were kept constant at their nominal values at all times. Several runs of both the F-H/OBE algorithm and the D-H/OBE algorithm were performed with different input and noise sequences. It was observed that the true parameter vector was sometimes out of the bounding ellipsoid at $t = 400$ and would be recaptured by the bounding ellipsoid after some number of samples (usually less than 50) thus verifying the claims made at the end of Section 3. It was also observed that the jump causes the resulting bounding ellipsoids to have smaller sizes. Intuitively, a jump at time t causes the set S_i, $i \geq t$, to have a smaller intersection with E_{i-1} and so the ellipsoid which bounds the intersection is also smaller. The parameter estimates (the centers of the bounding ellipsoids) are plotted against the true parameters in Figs. 2(a) and 2(b).

For the purpose of comparisons, Figs. 3(a) and 3(b) are results of implementing the RLS algorithm with an exponential forgetting factor for Case 1 and Case 2, respectively. The forgetting factor was set to be 0.99. It is clear that both of the OBE algorithms track better than the RLS algorithm.

The rescue schemes are also studied for both the F-H/OBE and D-H/OBE algorithms. The Rao-Huang rescue procedure [13] was employed for the D-H/OBE algorithm and yielded remarkable results. Another rescue scheme was implemented for the F-H/OBE algorithm by increasing the value of $\sigma^2(t-1)$ by an amount of $C + \frac{\delta}{\|x(t)\|}$ whenever $\sigma^2(t)$ becomes negative, where C is a positive constant. Note that the quantity $\delta / \| x(t) \|$ is the distance between the center of the ellipsoid E_{t-1}, i.e., $\theta(t-1)$, and the bisecting hyperplane of the observation set S_t, Eq.(2.4). From Figs. 4(a) and 4(b), it is seen clearly that the parameters are tracked extremely rapidly with these rescue schemes. Note that for Fig. 4(a), C is chosen to be 4.

6 Conclusions

A unified formulation of the set of OBE algorithms is presented and tracking characteristics of the methods are examined analytically. Simulation results for two OBE algorithms demonstrate their superior tracking properties. In cases of large parameter jumps, the automatic activation of a rescue procedure causes the parameters to be tracked extremely rapidly.

References

[1] B. Widrow et al., "Stationary and non-stationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151-1162, 1976.

[2] A. Benveniste and G. Ruget, "A measure of the tracking capability of recursive stochastic algorithms with constant gains," *IEEE Trans. Automat. Contr.*, vol. 27, pp. 639-649, 1982.

[3] E. Eleftheriou and D.D. Falconer, "Tracking properties and steady-state performance of RLS adaptive filter algorithms", *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 34, no. 5, pp. 1097-1109, Oct. 1986.

[4] B.D. Rao and R. Peng, "Tracking characteristics of the constrained IIR adaptive notch filter", *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 36, no. 9, pp. 1466-1479, Sept. 1988.

[5] S. Gunnarsson and L. Ljung, "Frequency domain tracking characteristics of adaptive algorithms", *IEEE Trans. Acoust., Speech and Signal Process.*, vol. 37, no. 7, pp.1072-1089, July 1989.

[6] E. Fogel and Y.F. Huang, "On the value of information in system identification - Bounded noise case," *Automatica*, vol. 18, pp. 229-238, 1982.

[7] S. Dasgupta and Y.F. Huang, "Asymptotically convergent modified recursive least squares with data dependent updating and forgetting factor for systems with bounded noise," *IEEE Trans. Information Theory*, vol. 33, pp. 383-392, 1987.

[8] J.R. Deller, Jr., "Set-membership identification in digital signal processing," *IEEE ASSP Magazine*, vol. 6, pp. 4-22, Oct. 1989.

[9] J.R. Deller, M. Nayeri, and S.F. Odeh, "Least square identification with error bounds," *Proc. IEEE* (in review).

[10] E. Walter and H. Piet-Lahanier, "Estimation of parameter bounds from bounded-error data: A survey," *Mathematics and Computers in Simulation*, vol. 32, pp. 449-468, 1990.

[11] J.R. Deller and S.F. Odeh, "SM-WRLS algorithms with an efficient test for innovation," *Proc. 9th IFAC / IFORS Symp. on Identification and System Parameter Estimation*, vol. 2, pp. 1044-1049, Budapest, July 1991.

[12] A.K. Rao, *Membership-Set Parameter Estimation via Optimal Bounding Ellipsoids* (Ph.D. Dissertation), University of Notre Dame, 1990.

[13] A.K. Rao and Y.F. Huang, "Tracking characteristics of an OBE parameter estimation algorithm", *IEEE Trans. Signal Processing*, vol. 41, no. 3, March 1993.

[14] J.R. Deller, Jr., M. Nayeri, and M.S. Liu, "Unifying the landmark developments in OBE processing," *Int. J. Automatic Control and Signal Process.* (in review).

[15] J.P. Norton and S.H. Mo, "Parameter bounding for time-varying systems," *Mathematics and Computers in Simulation*, vol. 32, pp. 527-534, 1990.

[16] J.R. Deller, Jr. and T.C. Luk, "Linear prediction analysis of speech based on set-membership theory," *Computer Speech and Language*, vol. 3, pp. 301-327, Oct. 1989.

Table 1: Specification of Existing OBE Algorithms

Algorithm	$\alpha_t(\lambda_t)$	$\beta_t(\lambda_t)$	Optimization
F-H/OBE[a]	$1/\sigma^2(t-1)$	λ_t/γ	$\mu_{vol}(t)$ or $\mu_{tr}(t)$
SM-WRLS[b]	1	λ_t	$\mu_{vol}(t)$ or $\mu_{tr}(t)$
Dual SM-WRLS[c]	λ_t	1	$\mu_{vol}(t)$ or $\mu_{tr}(t)$
D-H/OBE[d]	$1-\lambda_t$	λ_t	$\sigma^2(t)$
SM-SA[e]	$\Lambda^*_{t-1}/(\Lambda^*_{t-1}+\lambda_t)$ $\Lambda^*_{t-1} \overset{\text{def}}{=} \sum_{\tau=1}^{t-1}\lambda^*_\tau$	$\lambda_t/(\Lambda^*_{t-1}+\lambda_t)$	$\mu_{vol}(t)$ or $\mu_{tr}(t)$

Algorithm names and references: [a]Fogel-Huang OBE [6]; [b]set-membership-weighted recursive least squares [16]; [c]Dual-SM-WRLS [9,14]; [d]Dasgupta-Huang OBE [7]; [e]set-membership–stochastic approximation [9,14].

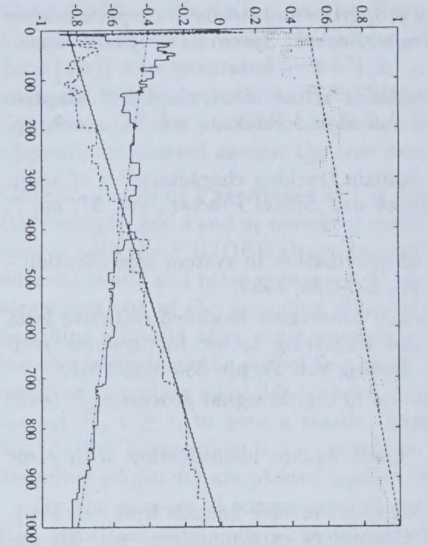


Figure 1(a) Tracking of slowly time-varying parameters by F-H/OBE.

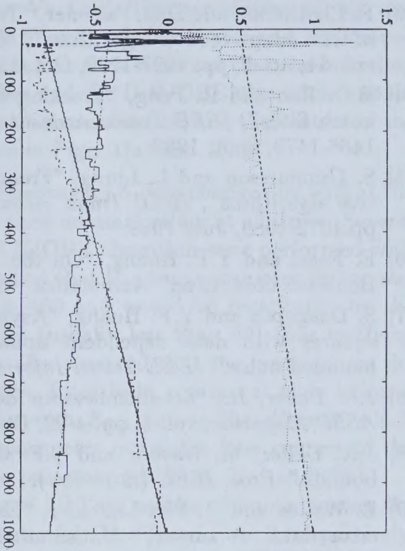


Figure 1(b) Tracking of slowly time-varying parameters by D-H/OBE.

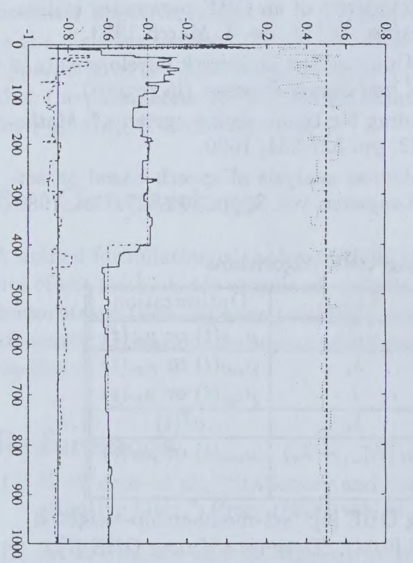


Figure 1(a) Tracking of slowly time-varying parameters by F-H/OBE.

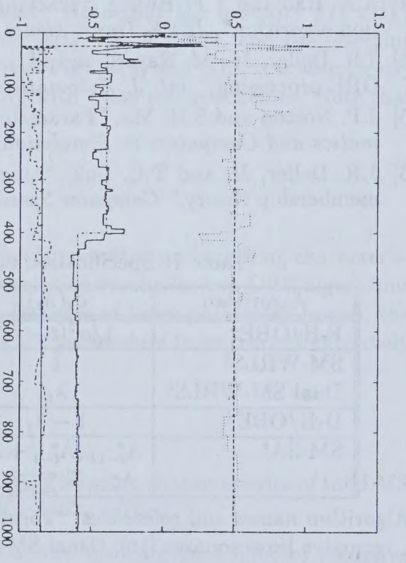


Figure 2(b) Tracking of jump parameters by D-H/OBE.

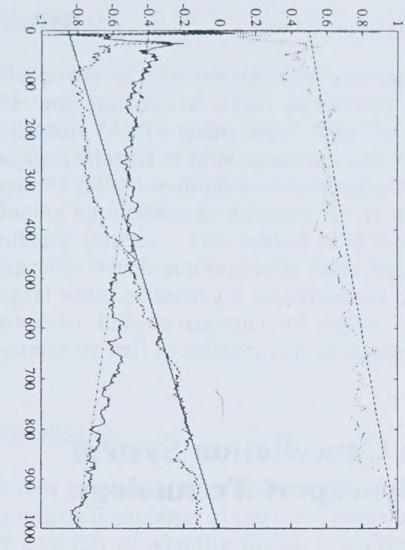


Figure 3(a) Tracking of slowly time-varying parameters by RLS with forgetting factor 0.99.

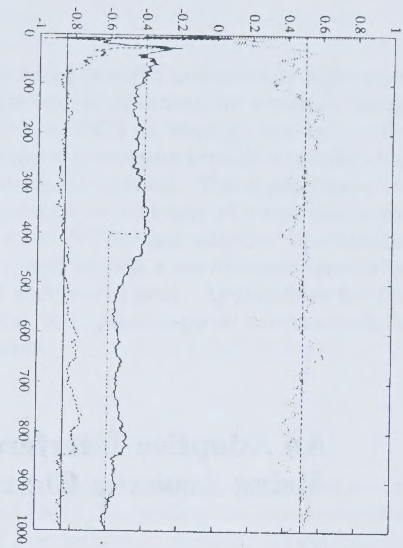


Figure 3(b) Tracking of jump parameters by RLS with forgetting factor 0.99.

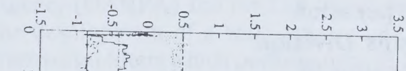


Figure 4(a) Tracking of jump parameters by F-H/OBE with a rescue scheme.

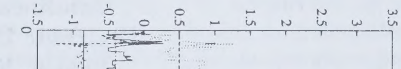


Figure 4(b) Tracking of jump parameters by D-H/OBE with a rescue scheme.

An Adaptive Interference Cancellation System Using Acoustic Charge Transport Technology

D.A. Fleisch, A.J. Vigil, A.W. Hull
and B.J. Hunsinger

Comlinear Corporation
Electronic Decisions Division
1776 E. Washington St.
Urbana, IL 61801

Abstract

The power of adaptive signal processing is now being brought to extremely high-speed RF and microwave signal processors through recent advances in acoustic charge transport (ACT) technology. The development of ACT technology has led to the implementation of fully programmable analog signal processors capable of performing over 45 billion multiply-and-accumulate operations per second. These processors are finding application as adaptive filters and equalizers in a variety of commercial and military systems. The subject of this paper is an ACT-based adaptive interference canceller which automatically identifies, tracks, and cancels a narrowband interfering signal while extracting a simultaneous desired wideband signal. Applications for this canceller include commercial digital broadcast and point-to-point communications systems as well as military spread-spectrum radios.

Introduction

A new generation of programmable high-speed signal-processing devices based on the emerging technology of acoustic charge transport (ACT) is enabling the implementation of a variety of adaptive filters, equalizers, and interference cancellers. These devices are finding application in military and commercial communications systems, as well as radar, electronic warfare, and test instrumentation systems.

The first successful demonstration of ACT technology was performed by Hunsinger and Hoskins at the University of Illinois in 1982 [1,2]. Subsequent development of this dual-use technology has been funded by a coalition of government, academic, and industrial sponsors, with major support provided by the Defense Advanced Research Projects Agency (DARPA) and the U.S. Air Force's Rome Laboratory. This support has led to the development of a wide range of ACT signal processors, including programmable transversal filters which perform over 45 billion multiply-and-accumulate operations per second [3-5]. A brief description of the operation of the ACT device and its use as a programmable transversal filter is presented in the following section; the use of ACT technology in adaptive interference cancellation is then described.

ACT Background

In its simplest form, an ACT device is an analog delay line in which discrete samples of an input signal are formed as a series of charge packets that propagate in a depleted transport channel induced in a GaAs substrate, as shown in Figure 1. These charge packets are formed and transported by electric fields that arise from piezoelectric coupling to a propagating surface acoustic wave (SAW) which is generated directly on the piezoelectric GaAs substrate. Unlike the operation of SAW devices, the ACT device employs the SAW only as a parametric pump or "clock"; all signal information is contained in the propagating charge packets rather than in the SAW itself. This obviates the deleterious effects of various acoustic phenomena which are observed in SAW devices, since these effects do not directly interact with the charge packets.

The charge packets are confined within a transport channel formed in an epitaxial surface layer having a thickness that is an appreciable fraction of the SAW wavelength at the clock frequency. This architecture isolates the charge packets from interfacial traps at the substrate surface and epitaxial layer interface. Thus, ACT devices operate in a deep "buried-channel" mode, and exhibit extremely high transport efficiencies at clock frequencies that are easily achieved in the UHF region (300 to 1000 MHz). The excellent transport efficiency at UHF clock rates forms the basis for a variety of high speed, high performance analog signal processors using ACT technology.

Other technical features that enhance the power of ACT devices in analog signal processors include:

- Intrinsic high performance sampling:

  The phenomenon leading to charge packet formation in the input section of an ACT device results in an intrinsically high performance sampling operation. Effective aperture times have been measured to be less than ten percent of a clock period leading to precise sample formation and the potential for providing frequency conversion directly in the ACT device.

- Non-destructive charge sensing:

  An image charge is induced in arbitrary metal features fabricated within the propagation path that allows the amount of charge in each packet to be sensed without affecting the contents of the packet. This feature enables the realization of high performance tapped delay line structures that form the basis for a rich set of analog signal processors.

- Integration with GaAs ICs:

  Although the fabrication of ACT devices requires a number of unique processing steps, the process is amenable to the integration of GaAs integrated circuit elements to provide interface and control functions. The potential for providing delay and completely integrated RF signal processing functions is unprecedented.

ACT Programmable Transversal Filter

One of the most useful applications of the delay and sensing functions inherent to the ACT device is in the transversal filter. The classical transversal filter is shown in Figure 2. In this device, an input signal is processed by passing it through a sequence of delay elements. After each element, the signal is sensed, weighted by a predetermined coefficient, and sent to an accumulator. The original signal is passed through another delay element, a different weighting coefficient is applied, and the result is sent to the accumulator. This cycle of delay, sense, weight, and accumulate is repeated in each stage of the filter. When the signs of the weighting coefficients match the phase of the input signal, the signals from each tap add coherently, and a large signal appears on the output terminal of the filter. The time-domain response of the filter is given by:

$$h(t) = \sum_{n=1}^{N} h_n \delta(t-nT)$$

where N represents the number of taps and T the delay between taps. The frequency-domain response of the transversal filter is:

$$H(\omega) = \sum_{n=1}^{N} h_n e^{-jn\omega T}$$

The ACT programmable transversal filter uses an ACT tapped delay line to accomplish the delay and sensing operations required by the transversal filter. In this device, a series of non-destructive sensing electrodes are used to sense the input signal as it propagates down the delay line. Each sensing tap is connected to a common summing bus (the accumulator of the transversal filter) through its own programmable attenuator. The values of the attenuators (the weighting coefficients) are set by a digital controller. All tap weights are stored in random access memory (RAM) which is monolithically integrated with the ACT delay line. A diagram of the ACT programmable transversal filter is shown in Figure 3.

ACT Adaptive Interference Cancellers

The ACT adaptive interference canceller tracks and cancels narrowband and frequency-modulated (FM) signals without affecting desired wideband signals occupying the same spectrum. In hostile electromagnetic environments, this canceller may be used to suppress spot jammers and frequency-hop jammers to permit detection and recovery of desired signals. In the presence of existing friendly FM transmissions, cancellation enables multiple signals to occupy the same spectrum.

Adaptive filters are employed for signal processing applications in which the design of a filter or equalizer cannot be determined a priori. An adaptive algorithm tunes the device parameters by minimizing a selected error measure. An adaptive interference canceller removes undesired frequency components from a received signal while passing the desired information.

There are two operations fundamental to an adaptive filter: convolution to compute the output of the adaptive filter, and filter coefficient update to compute the desired filter parameters. The convolution operation for wideband applications can require tens of billions of multiply-and-accumulate operations per second (MACS), which is considerably beyond the capability of today's digital signal processors. The ability of the ACT PTF to perform over 45 billion MACs allows this technology to overcome the computational "bottleneck" of digital adaptive filters. Digital logic may be employed to perform the adaptive tap updating for maximum accuracy and flexibility.

The conceptual block diagram of an ACT-based interference and noise cancellation system is shown in Figure 4. The incoming signal containing both the desired and undesired frequency components is split and sent to a summer through a direct path as well as a delayed path. The delay in the second path is made sufficiently long so as to allow the broadband signal components to decorrelate prior to summation with the direct path. The response of the ACT programmable transversal filter is then dynamically adjusted to minimize the error signal. In the interference canceller configuration, the system output is taken after summation; the periodic interference has been minimized by the adaptation process. In the noise canceller configuration, the system output is taken from the output of the PTF; the broadband component of the incoming signal has been minimized at this point [6].

The operating frequencies of the ACT PTF render it most fit for use in the RF spectrum. Figure 5 demonstrates the ability of an ACT adaptive interference canceller to excise an undesired narrowband signal while allowing an underlying pulse to pass through. For many communications applications it is most convenient to operate the adaptive filter directly at the IF stage, which is well-suited to the frequency range of ACT technology. Areas of application then include channel equalization for microwave communications and multipath suppression in mobile communications, narrowband interference suppression in spread-spectrum communications, and beamforming for direction finding, signal detection and estimation. Further, the exceptional computational performance of ACT devices permit implementation of frequency-domain adaptive filters, which facilitate low speed digital updating of the adaptive parameters for very high order adaptive filters [7].

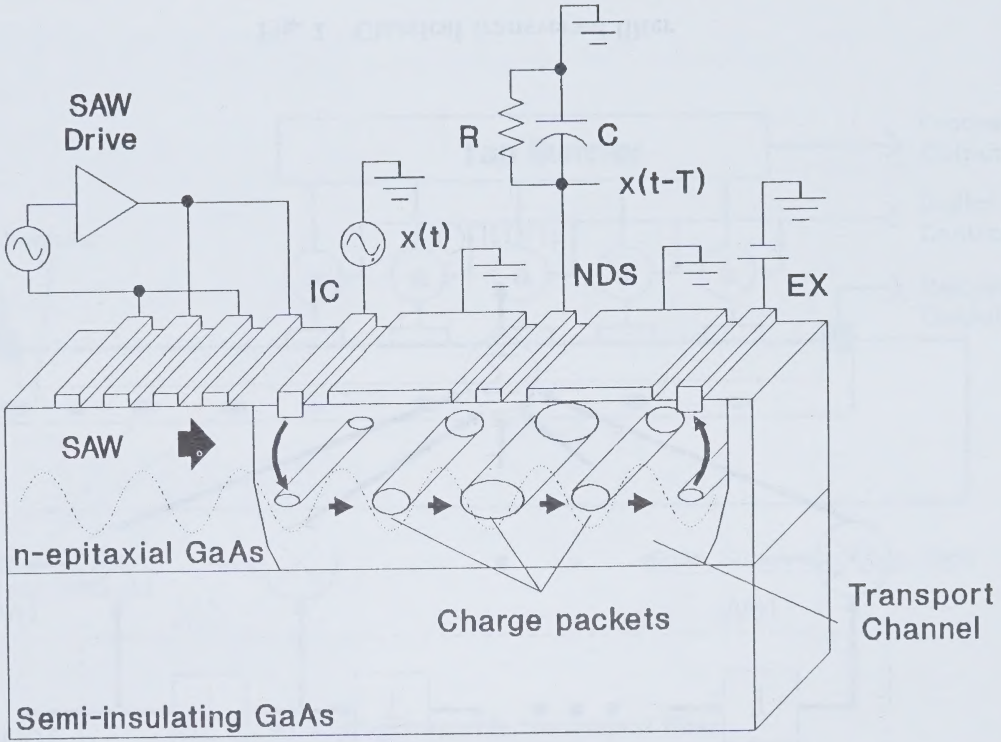

SAW
Drive
R
C
x(t-T)
x(t)
IC
NDS
EX
SAW
n-epitaxial GaAs
Charge packets
Transport
Channel
Semi-insulating GaAs

Fig. 1 ACT device architecture

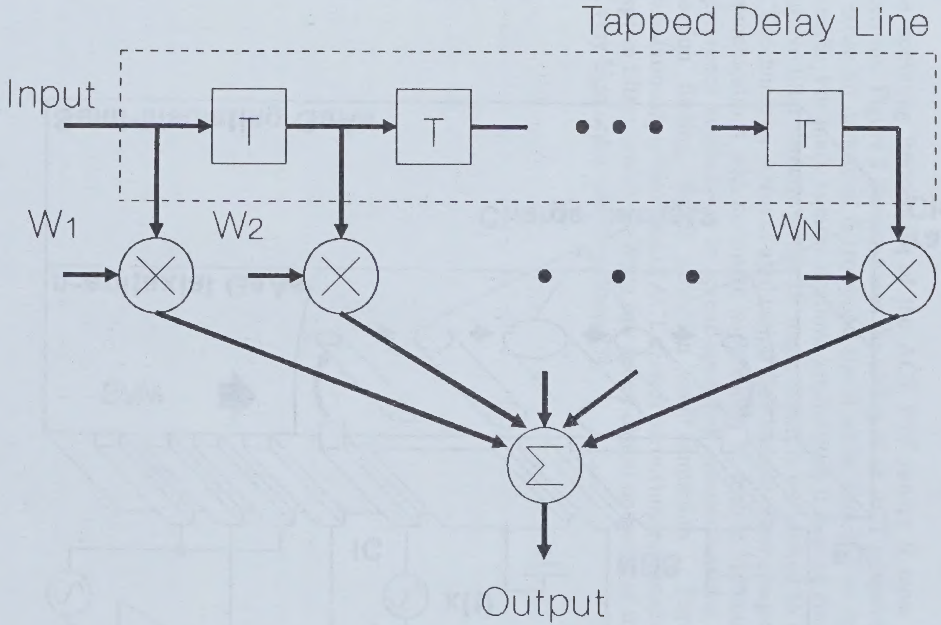

Tapped Delay Line
Input
T
T
T
W1
W2
WN
Σ
Output

Fig. 2 Classical transversal filter

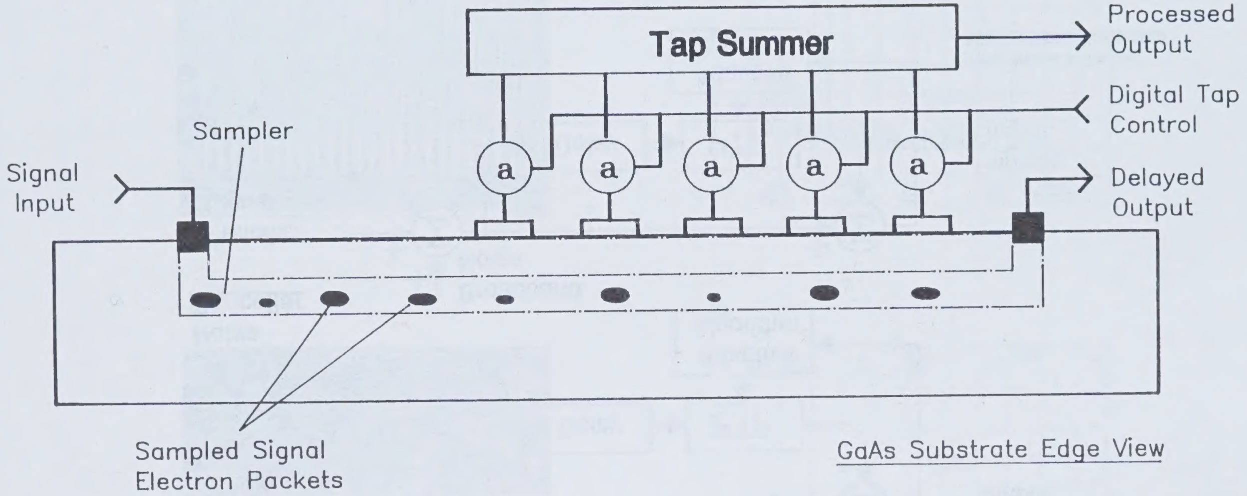
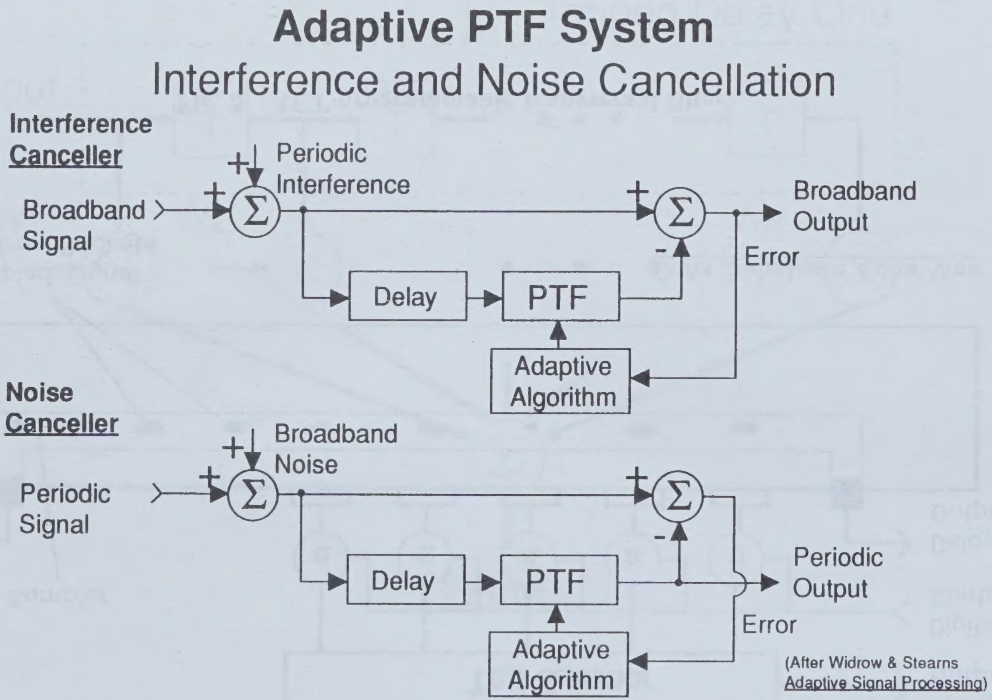

Tap Summer
Processed Output
Digital Tap Control
Delayed Output
Sampler
Signal Input
a
a
a
a
a
Sampled Signal Electron Packets
GaAs Substrate Edge View

Fig. 3 ACT programmable transversal filter


Adaptive PTF System
Interference and Noise Cancellation
Interference Canceller
Broadband Signal
+
+
Periodic Interference
Σ
Delay
PTF
Adaptive Algorithm
+
-
Σ
Broadband Output
Error
Noise Canceller
Periodic Signal
+
+
Broadband Noise
Σ
Delay
PTF
Adaptive Algorithm
+
-
Σ
Periodic Output
Error

(After Widrow & Stearns Adaptive Signal Processing)

Fig. 4 Adaptive PTF system: interference and noise cancellation

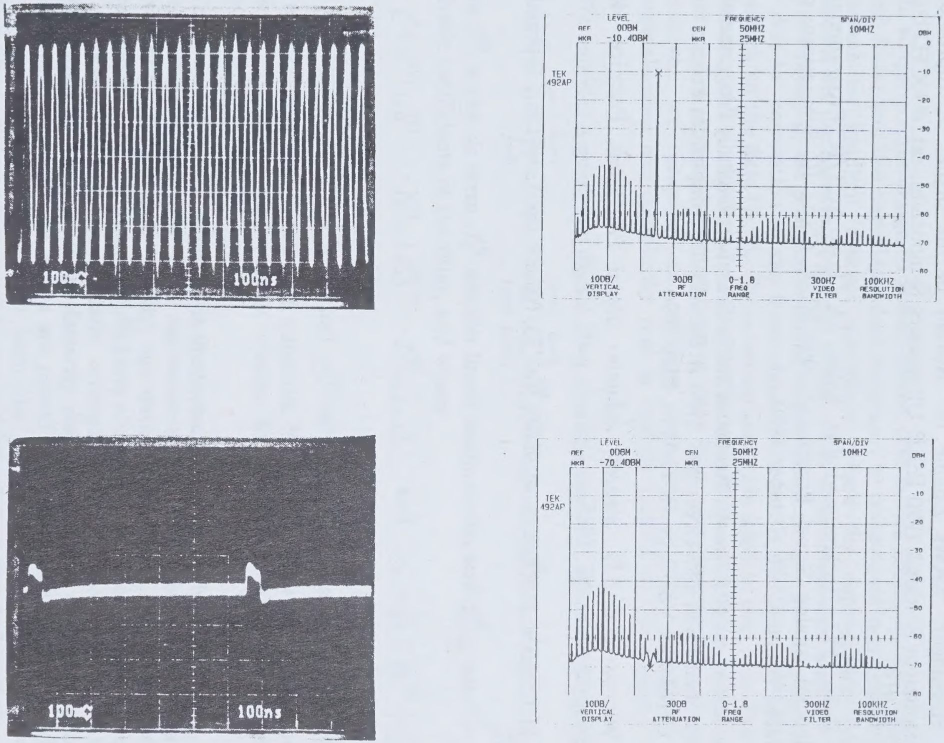


Fig. 5 Time and frequency response before and after cancellation

References

1. M.J. Hoskins and B.J. Hunsinger, "Simple Theory of Buried Channel Acoustic Charge Transport in GaAs," *Journal of Applied Physics*, Vol. 55, No. 2, pp. 413-426, January 1984.

2. M.J. Hoskins and B.J. Hunsinger, "Recent Developments in Acoustic Charge Transport Devices," 1986 IEEE Ultrasonics Symposium Proceedings.

3. B.J. Hunsinger, "SAWs Enable the Advent of the Signal Microprocessor Development Station," 1990 IEEE Ultrasonics Symposium, cat. #90CH2938-9, pp. 225-229.

4. J.E. Bales, M.J. Hoskins and P.H. Sahm, "A GaAs ACT/IC Programmable Wideband Analog Signal Processor," 1990 IEEE GaAs IC Symposium, cat. #CH2889-4, pp. 23-26, October 1990.

5. R.W. Miller and R.J. Kansy, "Acoustic Charge Transport Digitally Programmable Transversal Filter Development," 1990 IEEE MTT-S Symposium Proceedings, cat. CH2848-0, Vol. 3, pp. 1111-1114, May 1990.

6. B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice Hall: Englewood Cliffs, NJ, 1985.

7. S.U.H. Qureshi, *Proceedings of IEEE*, Vol. 73, Issue 9, pp. 1349-1387, September 1985.

An Efficient Two-Stage Lattice Implementation of a Second Order Adaptive Volterra Filter[1]

Miroslav V. Dokic　　*Peter M. Clarkson*

Dept. of Electrical and Computer Engineering
Illinois Institute of Technology
IIT Center
Chicago, IL 60616

In adaptive filter design, it is well-known that lattice filters exhibit convergence that is both faster and less data-dependent compared to transversal implementations. This is important for gradient descent algorithms where the convergence rate is highly dependent on the eigenvalue spread in the input data correlation matrix. In nonlinear adaptive filtering, the problem is further exacerbated since eigenvalue spread occurs not only due to correlation in the input data itself, but is also induced by the nonlinear operations on the data [1],[2]. Some researchers have tried to overcome this problem by formulating the nonlinear adaptive filter as a multichannel linear lattice structure [3]. However, this approach suffers from a very high computational requirement. As an alternative, we propose an efficient two-stage lattice structure where the orthogonalization has complexity that is linearly related to the memory span of the filter.

We consider a quadratic digital filter characterized by input-output relation:

$$y_n = f^{(0)} + \sum_{j=0}^{L-1} f_j^{(1)} x_{n-j} + \sum_{j_1=0}^{L-1} \sum_{j_2=j_1}^{L-1} f_{j_1,j_2}^{(2)} x_{n-j_1} x_{n-j_2} = \underline{f}^t \underline{x}_n ,$$

where $f^{(0)}$ is the *dc* term, $f_j^{(1)}$ are the linear components, and $f_{j_1,j_2}^{(2)}$ are the $L(L+1)/2$ quadratic coefficients of the filter, and where

$$\underline{f}^t = [f^{(0)} \mid f_0^{(1)} \cdots f_{L-1}^{(1)} \mid f_{0,0}^{(2)} \cdots f_{L-1,L-1}^{(2)}] \quad \text{and} \quad \underline{x}_n^t = [1 \mid \underline{x}_n^{(1)t} \mid \underline{x}_n^{(2)t}] ,$$

with

$$\underline{x}_n^{(1)t} = [x_n \; x_{n-1} \; \cdots \; x_{n-L+1}] \quad \text{and} \quad \underline{x}_n^{(2)t} = [x_n^2 \; x_n x_{n-1} \; \cdots \; x_n x_{n-L+1} \; \cdots \; x_{n-L+1}^2] .$$

The restriction of both linear and quadratic 'lengths' to L is for convenience, and the limits on j_1 and j_2 have been chosen to avoid multiplicities in the quadratic components.

The proposed lattice structure is illustrated in Figure 1. In the diagram, each block labeled 'lattice preprocessor' is a conventional FIR predictive lattice. The processing involves two lattice stages. Firstly, the data are input to a conventional predictive lattice structure with reflection coefficients chosen to minimize the mean-squared error. This processor produces a mutually orthogonal set of backward prediction errors, thereby removing the eigenvalue disparity due to the input data correlation [4]. The output backward prediction errors are passed to a linear adaptive filter. These same data are nonlinearly combined to form the inputs for the quadratic section of the adaptive filter. This nonlinear combining is followed by a reordering of the data. This reordering is the key to the efficiency gains in the proposed filter. We observe that for the nonlinear correlation matrix, assuming zero mean independent identically distributed (iid) data, non-zero off-diagonal elements occur only in those rows and columns whose corresponding diagonal element is the fourth order moment $E\{x_n^4\}$. So, by appropriate reordering of the quadratic elements, we can render the nondiagonal submatrix more compact, reducing the dimensionality of the eigenvalue problem from $L(L+1)/2$ to L. In the overall algorithm, we apply this reordered set of data as inputs to a second predictive lattice filter. This lattice completes the data orthogonalization. It is easy to verify that the resulting set of backward prediction errors forms a mutually orthogonal set. Since this set of backward prediction errors spans the same space as the input data, we have effectively removed the eigenvalue spread due to the nonlinear data combination. This orthogonalized data completes the required input set for the quadratic filter.

---

[1] *This work was supported by the National Science Foundation under grant no. MIP-9102620.*

A general input signal also exhibits eigenvalue disparity due to data correlation. This correlation is removed via the action of the first lattice preprocessor. Note that in practice, after the input data has been processed by the first lattice structure, a power normalization can be applied to equalize convergence for each coefficient in the quadratic section. This normalization is useful because the power of the backward prediction errors is monotonically nonincreasing with lattice stage [4]. Under these conditions, the mean convergence of the overall adaptive second order filter is uniform and decoupled. Extensive simulations have confirmed that even for an iid input, the convergence of the transversal LMS filter coefficients is highly nonuniform. By comparison, the convergence of the coefficients after the data has been processed by the lattice structure is greatly improved.

References

[1] M.V.Dokic and P.M.Clarkson, "On the performance of a second order adaptive Volterra filter," *IEEE Transactions on Signal Processing*, Vol. SP-41, pp.tbd, 1993.

[2] P.M.Clarkson and M.V.Dokic, "Stability and convergence behavior of a second order LMS Volterra filter," *Electronic Letters*, Vol. 27, pp.441-443, 1991.

[3] V.J.Mathews, "Adaptive polynomial filters", *IEEE Signal Processing Magazine*, pp.10-26, July 1991.

[4] S.Haykin, *Adaptive Filter Theory*, Prentice Hall, 1986.

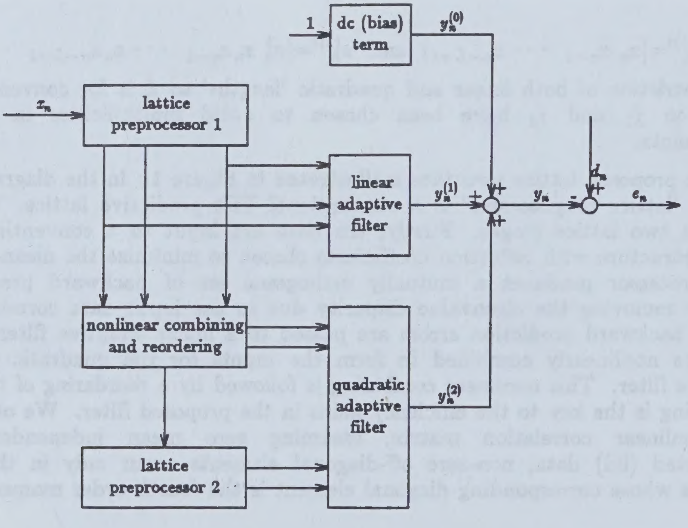


Figure 1. Block diagram of proposed two-stage lattice processor.

Positive Invariance of Symmetric Polyhedral Sets for Continuous-Time Linear Systems with Applications *

Derong Liu Anthony N. Michel
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

We first address the problem of positive invariance of symmetric polyhedral sets with respect to continuous-time linear time-invariant systems. Necessary and sufficient conditions for a symmetric polyhedral set to be positively invariant with respect to linear systems are established. We then establish a result on the robust positive invariance of symmetric polyhedra for the continuous-time linear systems with small perturbations and provide a solution to the so called linear constrained regulation problem–the stabilization of linear continuous-time systems under state and control constraints. The results established in this paper constitute continuous-time counterparts for the existing results for the discrete-time case.

I INTRODUCTION

In this paper, we first establish new results for symmetric polyhedral sets to be positively invariant with respect to systems given by

$$\dot{x}(t) = Ax(t), \quad t \geq 0, \tag{1}$$

where $x \in R^n$, $A = [a_{ij}] \in R^{n \times n}$.

---

*This work was supported by the National Science Foundation under grant ECS 91–07728.

A *state constraint set* Ω is said to be *positively invariant with respect to* (1), if for any initial point $x(0) \in \Omega$, the state of (1) satisfies $x(t) \in \Omega$ for all $t \geq 0$. The positive invariance of a state constraint set is crucial in state constrained control problems for linear systems. It is also true in practice that state constraint sets often appear to be of polyhedral type. A result on the positive invariance of symmetric polyhedral sets with respect to continuous-time linear systems is reported in [2]. In [2, Proposition 2], it is stated (without proof) that a symmetric polyhedral set given by

$$D(G,\rho) \triangleq \{x \in R^n : -\rho \leq Gx \leq \rho\},$$

(where $G \in R^{m\times n}$, $\rho \in R^m$, $\rho > 0$ and the inequality is applied componentwise) is positively invariant with respect to system (1), *if and only if* there exists a matrix $K = (k_{ij}) \in R^{m\times m}$, $k_{ii} \geq 0$, $i = 1, \cdots, m$, and a scalar $s_0 > 0$ such that

$$\begin{cases} (-s_0 I + K)G = GA \\ (-s_0 I + |K|)\rho \leq 0 \end{cases}, \tag{2}$$

where I represents the identity matrix and the absolute value is taken componentwise. (The condition $k_{ii} \geq 0$, $i = 1, \cdots, m$, is actually not specified in [2, Proposition 2]; however, as will be seen, it is a requirement.)

The present work is motivated by some of our recent results [5] which suggest necessary and sufficient conditions for a special case of polyhedral set, the unit hypercube, given by $D^n = \{x \in R^n \colon -1 \leq x_i \leq 1\}$, to be positively invariant with respect to system (1).

To arrive at our results, we first establish conditions under which $D^n = D(I, e_n)$ is positively invariant with respect to (1), where $e_n = (1, \cdots, 1)^T \in R^n$. We then develop results for the positive invariance of $D(I,\rho)$ and $D(G,\rho)$ with respect to (1), respectively.

Finally, we utilize the above results to (*i*) establish conditions for the robust positive invariance of symmetric polyhedra for linear systems with perturbations; and (*ii*) provide a solution to the continuous-time linear constrained regulation problem (with symmetric polyhedral state constraints).

II MAIN RESULTS

In [5], we proved that $\mu_\infty(A) < 0$ constitutes a *sufficient* condition for D^n to be positively invariant with respect to (1), where $\mu_\infty(\cdot)$ is the matrix measure induced by the matrix norm $\|\cdot\|_\infty$ and is given by

$$\mu_\infty(A) = \max_{1\leq i\leq n}\{a_{ii} + \sum_{j=1, j\neq i}^{n} |a_{ij}|\}. \tag{3}$$

(For definitions and properties of vector and matrix norms, refer, e.g., to [6]). In the following result, we show that the less restrictive condition

$$\mu_\infty(A) \leq 0 \tag{4}$$

constitutes a *necessary and sufficient* condition for D^n to be positively invariant with respect to (1).

Theorem 1 The set $D^n = D(I, e_n) = \{x \in R^n : -e_n \leq x \leq e_n\}$ is positively invariant with respect to system (1) *if and only if*

$$\overline{A}e_n \leq 0, \tag{5}$$

where $\overline{A} = [\overline{a}_{ij}]$, $\overline{a}_{ij} = |a_{ij}|$ for all $i \neq j$, and $\overline{a}_{ii} = a_{ii}$ for $i = 1, \cdots, n$.

Proof: First, we note that (4) and (5) are equivalent conditions.

Sufficiency. The solution of (1) is given by $x(t) = e^{At}x(0)$ for $t \geq 0$, where $x(0) \in R^n$ is the initial condition, and $\|x(t)\|_\infty = \|e^{At}x(0)\|_\infty \leq \|e^{At}\|_\infty \|x(0)\|_\infty$. Using the fact that $\|e^{At}\|_\infty \leq e^{\mu_\infty(A)t}$ for all $t \geq 0$ (see [3], p.59), we have $\|x(t)\|_\infty \leq e^{\mu_\infty(A)t}\|x(0)\|_\infty \leq \|x(0)\|_\infty$ for all $t \geq 0$, since $\mu_\infty(A) \leq 0$. This in turn implies that $x(t) \in D^n$ *for all* $t \geq 0$, provided that $x(0) \in D^n$. Here we have used the fact that $x \in D^n$ *if and only if* $\|x\|_\infty \leq 1$.

Necessity. It suffices to show that when (5) does not hold, there always exist some initial point in D^n, for which $x(0) \in D^n$ does not imply that $x(t) \in D^n$ for all $t \geq 0$.

Let us assume that condition (5) does not hold for the k^{th} row of matrix A, *i.e.*,

$$a_{kk} + \sum_{j=1, j\neq k}^{n} |a_{kj}| = \varepsilon_k > 0.$$

From the solution of (1), we see that when $t > 0$ is small, we have $x(t) = [I + At + Q(t)]x(0)$, where $Q(t) = [q_{ij}(t)] \in R^{n\times n}$ satisfies

$$\lim_{t\to 0^+} \frac{Q(t)}{t} = 0. \tag{6}$$

Choose the initial condition $x(0) = [\text{sign}(a_{k1}), \cdots, \text{sign}(a_{k,k-1}), 1, \text{sign}(a_{k,k+1}), \cdots, \text{sign}(a_{kn})]^T$. Then, $x(0) \in D^n$ and the k^{th} component of $x(t)$ becomes

$$x_k(t) = 1 + (a_{kk} + \sum_{j=1, j\neq k}^{n} |a_{kj}|)t + \theta_k(t) = 1 + \varepsilon_k t + \theta_k(t) \tag{7}$$

where

$$\theta_k(t) = q_{kk}(t) + \sum_{j=1, j\neq k}^{n} q_{kj}(t)\mathrm{sign}(a_{kj}).$$

From (6) we know that for any $\varepsilon > 0$, there exists a $\delta > 0$, such that $|\theta_k(t)|/t < \varepsilon$ when $0 < t < \delta$. Choosing $\varepsilon = \varepsilon_k$, then, there exists a $\delta_k > 0$, such that $|\theta_k(t)|/t < \varepsilon_k$ when $0 < t < \delta_k$, *i.e.*, $\varepsilon_k t > |\theta_k(t)| \geq -\theta_k(t)$ when $0 < t < \delta_k$. This in turn implies that $x_k(t) = 1 + \varepsilon_k t + \theta_k(t) > 1 = x_k(0)$ for $0 < t < \delta_k$, *i.e.*, for this specific choice of $x(0) \in D^n$, $x(t) \notin D^n$ for $0 < t < \delta_k$. This proves necessity. ∎

The next result generalizes Theorem 1 to the case of constraint sets being the hypercuboid, given by $D(I, \rho) = \{x \in R^n : -\rho \leq x \leq \rho\}$, where $\rho \in R^n$ and $\rho > 0$.

Theorem 2 The set $D(I, \rho)$ is positively invariant with respect to (1) *if and only if*

$$\overline{A}\rho \leq 0, \tag{8}$$

where $\rho = (\rho_1, \cdots, \rho_n)^T$, $\rho_i > 0$ for $i = 1, \cdots, n$, and $\overline{A}$ is defined in Theorem 1.

Proof: Let $T = \mathrm{diag}(\rho) = \mathrm{diag}(\rho_1, \cdots, \rho_n)$. Then, $-Te_n = -\rho \leq x \leq \rho = Te_n$ is equivalent to $-e_n \leq T^{-1}x \leq e_n$, since $\rho_i > 0$ for $i = 1, \cdots, n$.

Letting $y = T^{-1}x$, we have

$$\dot{y} = T^{-1}\dot{x} = T^{-1}Ax = T^{-1}ATy. \tag{9}$$

We claim that "$D(I, \rho)$ is positively invariant with respect to (1)" is equivalent to "$D(I, e_n)$ is positively invariant with respect to (9)". To see this, we note that the former is equivalent to $e^{At}x \in D(I, \rho)$ for all $t \geq 0$ and for any $x \in D(I, \rho)$, and the latter is equivalent to $e^{T^{-1}ATt}y \in D(I, e_n)$ for all $t \geq 0$ whenever $y \in D(I, e_n)$. Also, $x \in D(I, \rho)$ is equivalent to $y \in D(I, e_n)$ and $e^{At}x \in D(I, \rho)$ is equivalent to $e^{T^{-1}ATt}y \in D(I, e_n)$ (this is true because $e^{At}x \in D(I, \rho)$ is equivalent to $T^{-1}e^{At}x \in D(I, e_n)$ and $T^{-1}e^{At}x = T^{-1}e^{At}Ty = e^{T^{-1}ATt}y$).

From Theorem 1, we see that $D(I, e_n)$ is positively invariant with respect to (9) *if and only if*

$$\overline{T^{-1}AT}e_n \leq 0, \tag{10}$$

where $\overline{T^{-1}AT}$ is defined in a similar manner as $\overline{A}$ above. We now compute

$$T^{-1}AT = \begin{pmatrix} \rho_1^{-1} & 0 & \cdots & 0 \\ 0 & \rho_2^{-1} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \rho_n^{-1} \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} \rho_1 & 0 & \cdots & 0 \\ 0 & \rho_2 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \rho_n \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12}\rho_2\rho_1^{-1} & \cdots & a_{1n}\rho_n\rho_1^{-1} \\ a_{21}\rho_1\rho_2^{-1} & a_{22} & \cdots & a_{2n}\rho_n\rho_2^{-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1}\rho_1\rho_n^{-1} & a_{n2}\rho_2\rho_n^{-1} & \cdots & a_{nn} \end{pmatrix}.$$

From this we can see that (10) is equivalent to

$$a_{ii} + \sum_{j=1, j\neq i}^{n} |a_{ij}| \frac{\rho_j}{\rho_i} \leq 0, \ \text{ for } \ i = 1, \cdots, n,$$

or

$$a_{ii}\rho_i + \sum_{j=1, j\neq i}^{n} |a_{ij}| \rho_j \leq 0, \ \text{ for } \ i = 1, \cdots, n.$$

Thus, (10) and (8) are equivalent.

The above proves that (8) is a necessary and sufficient condition for the set $D(I, \rho)$ to be positively invariant with respect to system (1). ∎

The proof of Theorem 2 motivates the following lemma, which is required in the proof of our next result (Theorem 3).

Lemma 1 Let $GA = HG$, where $G \in R^{m \times n}$ and $H \in R^{m \times m}$. Then, the following two statements are equivalent:

(*i*) $D(G, \rho)$ is positively invariant with respect to (1);

(*ii*) $D(I, \rho)$ is positively invariant with respect to system (11) given by

$$\dot{y} = Hy, \tag{11}$$

where $y = Gx$.

Proof: (*i*) $\Rightarrow$ (*ii*). Suppose $y \in D(I, \rho)$. Since $y = Gx$, $y \in D(I, \rho)$ is equivalent to $Gx \in D(I, \rho) \Rightarrow -\rho \leq Gx \leq \rho \Rightarrow x \in D(G, \rho) \Rightarrow -\rho \leq Ge^{At}x \leq \rho$ for all $t \geq 0$, since $D(G, \rho)$ is positively invariant with respect to (1) by assumption. $\Rightarrow -\rho \leq e^{Ht}Gx \leq \rho$, since $GA = HG$ implies that $Ge^{At} = e^{Ht}G$. $\Rightarrow -\rho \leq e^{Ht}y \leq \rho$ for all $t \geq 0$.

This proves that if $y \in D(I, \rho)$, then $e^{Ht}y \in D(I, \rho)$ for all $t \geq 0$. Therefore, $D(I, \rho)$ is positively invariant with respect to (11) *if* $D(G, \rho)$ is positively invariant with respect to system (1).

(*ii*) $\Rightarrow$ (*i*). Suppose $x \in D(G, \rho)$ and $D(I, \rho)$ is positively invariant with respect to system (11). Then $-\rho \leq Gx \leq \rho \Rightarrow -\rho \leq y \leq \rho \Rightarrow -\rho \leq e^{Ht}y \leq \rho$ for all $t \geq 0$ $\Rightarrow -\rho \leq e^{Ht}Gx \leq \rho$ for all $t \geq 0 \Rightarrow -\rho \leq Ge^{At}x \leq \rho$ for all $t \geq 0 \Rightarrow D(G, \rho)$ is positively invariant with respect to (1). ∎

The following lemma is also required in the proof of Theorem 3.

Lemma 2 If the set $D(G, \rho)$ is positively invariant with respect to (1), then there always exists a matrix $H \in R^{m \times m}$ such that $GA = HG$.

Proof: If $D(G, \rho)$ is positively invariant with respect to (1), $D(G, c\rho)$ is also positively invariant with respect to (1) for any scalar $c > 0$. This can be seen from the fact that if $|Gx| \leq \rho$ implies $|Ge^{At}x| \leq \rho$ for all $t \geq 0$, then $|Gx| \leq c\rho$ implies $|Ge^{At}x| \leq c\rho$ for all $t \geq 0$ and for any $c > 0$. We next prove that $Gx = 0$ implies that $Ge^{At}x = 0$ for all $t \geq 0$.

Suppose that $Gx_0 = 0$ but $Ge^{At}x_0 \neq 0$ for some $t > 0$. Let

$$c = \max_{t>0}\left\{\frac{|(Ge^{At}x_0)_i|}{\rho_i},\ 1 \leq i \leq n\right\} > 0,$$

where $(Ge^{At}x_0)_i$ denotes the i^{th} component of $Ge^{At}x_0$. Then we arrive at the contradiction that $x_0 \in D(G, \frac{c}{2}\rho)$ but $e^{At}x_0 \notin D(G, \frac{c}{2}\rho)$ for some $t > 0$. Therefore, $Gx = 0$ implies $Ge^{At}x = 0$ for all $t \geq 0$.

It now follows that the rows of Ge^{At} belong to the subspace in R^n which is spanned by the rows of G and which is orthogonal to the null space of G. It also follows that every row of Ge^{At} can be represented by a linear combination of the rows of G. Thus, $(Ge^{At})_i = f_{i1}(t)(G)_1 + \cdots + f_{im}(t)(G)_m$, for $i = 1, \cdots, m$, *i.e.*,

$$Ge^{At} = F(t)G \text{ for all } t \geq 0, \tag{12}$$

where $(G)_i$ is the i^{th} row of G and $F(t) = [f_{ij}(t)] \in R^{m \times m}$.

Taking the time derivative on both sides of (12), we obtain

$$GAe^{At} = F'(t)G \text{ for all } t > 0. \tag{13}$$

This implies that $GA = HG$, where $H = F'(0^+) = \lim_{t \to 0^+} F'(t)$. Notice that $\lim_{t \to 0^+} F'(t)$ exists because, as $t \to 0^+$, the limit on the left hand side of (13) exists. ∎

We are now in a position to establish new conditions for symmetric polyhedral sets to be positively invariant with respect to continuous-time linear systems.

Theorem 3 The set $D(G, \rho) = \{x \in R^n : -\rho \leq Gx \leq \rho\}$ is positively invariant with respect to (1) *if and only if* there exists an $H = (h_{ij}) \in R^{m \times m}$, such that

$$\begin{cases} GA = HG \\ \overline{H}\rho \leq 0 \end{cases}, \tag{14}$$

where $\overline{H} = [\overline{h}_{ij}]$, $\overline{h}_{ij} = |h_{ij}|$ for all $i \neq j$, and $\overline{h}_{ii} = h_{ii}$ for $i = 1, \cdots, n$.

Proof: Sufficiency. Let $y = Gx$. Then

$$\dot{y} = G\dot{x} = GAx = HGx = Hy, \tag{15}$$

since we assume that there exists an $H \in R^{m\times m}$ such that $GA = HG$.

From Theorem 1, we know that $\overline{H}\rho \leq 0$ is a necessary and sufficient condition for $D(I, \rho)$ to be a positively invariant set with respect to (15). From Lemma 1, we see that $\overline{H}\rho \leq 0$ is also a necessary and sufficient condition for $D(G, \rho)$ to be a positively invariant set with respect to (1) when we assume that $GA = HG$.

Necessity. From Lemma 2 we see that if $D(G, \rho)$ is positively invariant with respect to (1), then there exists a matrix $H \in R^{m\times m}$ such that $GA = HG$. Letting $y = Gx$, we can see that (15) still holds. The rest of the proof follows directly from Lemma 1. ∎

Remark 1 Clearly, conditions (2) and (14) are equivalent when all the diagonal elements of matrix K in (2) are non-negative. Thus, $k_{ii} \geq 0$, $i = 1, \cdots, m$, is a requirement. ∎

Remark 2 The polyhedral set $D(G, \rho)$ is positively invariant with respect to discrete-time systems

$$x(k + 1) = Ax(k),\ k = 0, 1, \cdots, \tag{16}$$

if and only if there exists a matrix $H \in R^{m\times m}$ such that

$$GA = HG$$

and

$$|H|\rho \leq \rho.$$

This is obtained in [1], Proposition 2. Theorem 3 in the present paper constitutes a continuous-time counterpart for the discrete-time result obtained in [1]. It appears that results in this paper are presented in a form which is simpler and clearer than that stated in Equation (2). ∎

Our next result concerns the robust positive invariance of $D(G, \rho)$ with respect to system (1) when $G \in R^{n\times n}$ and $\text{rank}G = n$.

Theorem 4 Assume that for $D(G, \rho)$, $G \in R^{n\times n}$ and G has full rank. If there exists for system (1) an $H \in R^{n\times n}$ such that $HG = GA$ and $\overline{H}\rho < 0$, where $\overline{H}$ is defined in Theorem 3, then $D(G, \rho)$ is a positively invariant set of the perturbed systems

$$\dot{x} = (A + \Delta)x, \tag{17}$$

for all $\Delta \in R^{n\times n}$ such that $\|\Delta\|_\infty \le d$ for some $d > 0$. (The precise characterization of d will become clear in the proof.)

Proof: Let $T = \mathrm{diag}(1/\rho_i)G$, $H_0 = TAT^{-1}$, and $\nu = -\mu_\infty(H_0)$, where $\mathrm{diag}(1/\rho_i) = \mathrm{diag}(1/\rho_1, \cdots, 1/\rho_n)$. Then, $\nu > 0$ since

$$\begin{aligned} H_0 = TAT^{-1} &= \mathrm{diag}(1/\rho_i)GAT^{-1} \\ &= \mathrm{diag}(1/\rho_i)HG[\mathrm{diag}(1/\rho_i)G]^{-1} = \mathrm{diag}(1/\rho_i)H\mathrm{diag}(\rho_i) \end{aligned}$$

and

$$\begin{aligned} \mu_\infty(H_0) &= \max_{1\le i\le n}\{(\overline{H}_0 e_n)_i\} = \max_{1\le i\le n}\{[\mathrm{diag}(1/\rho_i)\overline{H}\mathrm{diag}(\rho_i)e_n]_i\} \\ &= \max_{1\le i\le n}\{[\mathrm{diag}(1/\rho_i)\overline{H}\rho]_i\} < 0. \end{aligned}$$

Note that in the above we have used the fact that $\rho_i > 0$ for $i = 1, \cdots, n$.

Define $H_\Delta = T(A+\Delta)T^{-1}$. Then

$$H_\Delta T = T(A+\Delta). \tag{18}$$

We now compute

$$\begin{aligned} \mu_\infty(H_\Delta) &= \mu_\infty(T(A+\Delta)T^{-1}) \\ &\le \mu_\infty(TAT^{-1}) + \mu_\infty(T\Delta T^{-1}) = -\nu + \mu_\infty(T\Delta T^{-1}), \end{aligned} \tag{19}$$

and

$$\mu_\infty(T\Delta T^{-1}) \le \|T\Delta T^{-1}\|_\infty \le \|T\|_\infty \|\Delta\|_\infty \|T^{-1}\|_\infty. \tag{20}$$

Thus, if we choose

$$d = \frac{\nu}{\|T\|_\infty \|T^{-1}\|_\infty}, \tag{21}$$

then

$$\mu_\infty(H_\Delta) \le 0, \tag{22}$$

provided that $\|\Delta\|_\infty \le d$. Note that in (21) $d > 0$ since $\nu > 0$. For the validity of (19) and (20) please refer to [3], p.41.

It is clear from (18) and (22) that $D(T, e_n)$ is positively invariant with respect to (17). Hence, $D(G,\rho)$ is positively invariant with respect to (17) because $D(G,\rho) = D(T, e_n)$. ■

From (21), we see that the bound d of the admissible perturbations for the coefficient matrix in system (17) will increase as $\mu_\infty(H_0)$ and $\|T\|_\infty \|T^{-1}\|_\infty$ decrease.

Remark 3 A similar result on the robust positive invariance of symmetric polyhedra for *discrete-time* linear systems was obtained in [4]. *We point out that the results presented in Theorem 4 can only be derived if one uses* (14), *in place of* (2), *to state necessary and sufficient conditions for a polyhedral set to be positively invariant with respect to continuous-time linear systems.* ∎

III APPLICATION TO LINEAR CONSTRAINED REGULATION PROBLEMS

In this section we present an application of the results given in Section 2.

Consider a continuous-time linear control system given by

$$\dot{x} = Ax + Bu \tag{23}$$

with state constraints $x \in D(G,\rho) = \{x \in R^n : = -\rho \leq Gx \leq \rho\}$ and control constraints $|u| \leq \eta$, where $A \in R^{n\times n}$, $B \in R^{n\times p}$, $G \in R^{m\times n}$, $\text{rank}G = m$, $\rho = (\rho_1, \cdots, \rho_m)^T$, $\rho_i > 0$ for $i = 1, \cdots, m$, $\eta = (\eta_1, \cdots, \eta_p)^T$, and $\eta_i > 0$ for $i = 1, \cdots, p$.

We wish to determine a linear state feedback law $u = Fx$ with $F \in R^{n\times p}$ such that for *all* initial states in $D(G,\rho)$, the states of system (23) are transferred to the origin asymptotically, with both states and control not violating their constraints. Such a problem is often referred to as a Linear Constrained Regulation Problem (LCRP) (cf., [7], [8]). Our next result provides a solution for the LCRP described above.

Corollary 1 If there exists an $F \in R^{n\times p}$ and an $H \in R^{m\times m}$ such that $A + BF$ is stable (*i.e.*, all eigenvalues of $A + BF$ have negative real parts), $GA + GBF = HG$, $\overline{H}\rho \leq 0$, and $D(G,\rho) \subseteq D(F,\eta)$, then the control law $u = Fx$ is a solution to the LCRP.

Proof: The proof follows directly from Theorem 3. ∎

Remark 4 The continuous-time LCRP was addressed in [7] when the state constraint set is a (nonsymmetric) closed convex polyhedral domain. We emphasize that the state constraint sets are symmetric and may be *unbounded* in our case. ∎

Remark 5 After some manipulations, the solutions for the LCRP presented in Corollary 1 can be obtained by linear programming as proposed for the discrete-time case in [8], *i.e.*, for an LCRP with A, B, G, ρ, and η given, the matrices F and H can be solved by linear programming. ∎

IV CONCLUDING REMARKS

We established in this paper a set of necessary and sufficient conditions for symmetric polyhedral sets to be positively invariant with respect to linear systems. Results for two special cases of polyhedral sets, namely, the hypercube and the hyper-cuboid, to be positively invariant with respect to linear systems were established as preliminaries. We also established an important result for the robust positive invariance of symmetric polyhedra with respect to continuous-time linear systems with perturbations using our above results. A straightforward application to the continuous-time linear constrained regulation problem was also provided in the present paper.

References

[1] G. Bitsoris, Positively Invariant Polyhedral Sets of Discrete-Time Linear Systems, *Int. J. Control* **47** (1988) 1713–1726

[2] E. B. Castelan, J. C. Hennet, On Invariant Polyhedra of Continuous-Time Systems, *Proceedings of the 30th Conference on Decision and Control,* Brighton, England (1991) 1736–1741

[3] W. A. Coppel, *Stability and Asymptotic Behavior of Differential Equations* (D. C. Heath and Company, Boston, MA, 1965)

[4] J. C. Hennet, E. B. Castelan, Robust Invariant Controllers for Constrained Linear Systems, *Proc. of 1992 American Control Conference,* Chicago, IL (1992) 993–997

[5] Derong Liu, A. N. Michel, Asymptotic Stability of Systems Operating on a Closed Hypercube, to appear in *Systems & Control Letters* **19** (1992)

[6] A. N. Michel, C. J. Herget, *Mathematical Foundations in Engineering and Science: Algebra and Analysis* (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981)

[7] M. Vassilaki, G. Bitsoris, Constrained Regulation of Linear Continuous-Time Dynamical Systems, *Systems & Control Letters* **13** (1989) 247–252

[8] M. Vassilaki, J. C. Hennet, G. Bitsoris, Feedback Control of Linear Discrete-Time Systems Under State and Control Constraints, *Int. J. Control* **47** (1988) 1727–1735

ON THE INVARIANCE OF CONTROLLABILITY INDICES OF LINEAR MIMO SYSTEMS

S. Bingulac and H. F. VanLandingham

Bradley Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Abstract

Contrary to the results indicated in the literature, it is found that the set of controllability indices for a linear MIMO system is *not* invariant with respect to the ordering of the inputs. Conditions specifying when the set of controllability indices of a linear MIMO pair $\{\mathbf{A},\mathbf{B}\}$ is invariant under any ordering of columns of input matrix $\mathbf{B}$ are given, thereby correcting an oversight committed in references [1] and [2].

Introduction

Consider a pair $\{\mathbf{A},\mathbf{B}\}$, $\mathbf{A}$ an $n \times n$ and $\mathbf{B}$ an $n \times m$ matrix, respectively, with $m > 1$. The $[n \times (n-m+1)m]$ controllability matrix $\mathbf{Q}_c$ of $\{\mathbf{A},\mathbf{B}\}$ is defined by

$$\mathbf{Q}_c = \left[\mathbf{b}_1 \;\; \mathbf{b}_2 \;\cdots\; \mathbf{b}_m \;\; \mathbf{A}\mathbf{b}_1 \;\; \mathbf{A}\mathbf{b}_2 \;\cdots\; \mathbf{A}\mathbf{b}_m \;\; \mathbf{A}^2\mathbf{b}_1 \;\;\cdots\;\; \mathbf{A}^{n-m}\mathbf{b}_m\right]$$

If the pair $\{\mathbf{A},\mathbf{B}\}$ is controllable, then by checking linear independence of the columns of $\mathbf{Q}_c$ "from the left," it is possible to select n linearly *independent* columns. Let $\mathbf{Q}_c^*$ contain exactly n linearly independent columns taken from $\mathbf{Q}_c$ in the order as they are selected from $\mathbf{Q}_c$. The selection of columns from $\mathbf{Q}_c$ into $\mathbf{Q}_c^*$ is done according to *Scheme II*, see [2].

Associate with $\mathbf{Q}_c$ the $(n-m+1)m$-dimensional "selector" row $\mathbf{r}$ containing unity elements at the $[i+(j-1)m]$th location, for $j=[0,n-m]$ and $i=[1,m]$, corresponding to the column $\mathbf{A}^j\mathbf{b}_i$ of $\mathbf{Q}_c$, if this column is selected in $\mathbf{Q}_c^*$; otherwise row $\mathbf{r}$ contains zero. The total number of unities in $\mathbf{r}$ is equal to n.

There are μ_i columns in $\mathbf{Q}_c^*$ "associated with" columns $\mathbf{b}_i$, $i=[1,m]$, of the form

$$\mathbf{b}_i,\; \mathbf{A}\mathbf{b}_i,\; \cdots,\; \mathbf{A}^{\mu_i-1}\mathbf{b}_i$$

The set $\mu = \{\mu_1, \ldots, \mu_m\} = \{\ \mu_i\ \}$ is referred to as the set of *controllability indices of* $\{\mathbf{A},\mathbf{B}\}$, [1], [2]. It is straightforward to show that the set μ is invariant under any similarity transformation, [1, Chen, page 191, Theorem 5-8]. In that Theorem it is also stated that the set μ is invariant under any ordering of columns of $\mathbf{B}$. A similar statement can also be found in [2, Kailath, page 428]. The purpose of this paper is to show that this is not always true. To this end, in addition to the concepts of controllability matrix and controllability indices, we need the following two definitions:

Definition 1.

The set of controllability indices $\mu = \{\mu_1, \ldots, \mu_m\}$ is *generic* if the values $\{\mu_i\}$ satisfy:

$$\mu_i = \left\{ \begin{array}{ll} \left[\dfrac{n}{m}\right] + 1 & \text{for } i = [1, s] \\ \left[\dfrac{n}{m}\right] & \text{for } i = [s+1, m] \end{array} \right\} \tag{1}$$

where $[x]$ denotes integer part of x, while s is given by:

$$s = n - \left[\frac{n}{m}\right] m \tag{2}$$

Note that if m divides n without reminder, then $s = 0$, and all the values μ_i are equal to n/m.

The above set is referred to as *generic* because when elements of matrices in a pair $\{\mathbf{A},\mathbf{B}\}$ are selected "at random," the controllability indices of $\{\mathbf{A},\mathbf{B}\}$ are "almost always" given by Eq.(1).

Definition 2.

In the case that the set μ of controllability indices of a given $\{\mathbf{A},\mathbf{B}\}$ is generic, the controllability matrix $\mathbf{Q}_c$ and the elements of the row $\mathbf{r}$ are:

$$\begin{array}{l} \mathbf{Q}_c = [\ \mathbf{B} \quad \mathbf{AB} \quad \mathbf{A}^2\mathbf{B} \quad \cdots \quad \mathbf{A}^{n-m}\mathbf{B}\ ] \\ \mathbf{r} = [\ 1 \cdots 1 \quad 1 \cdots 1 \quad 1 \cdots 1 \quad 0 \cdots 0\ ] \end{array} \tag{3}$$

where the unit elements of $\mathbf{r}$ correspond to the first n columns of $\mathbf{Q}_c$.

If $s > 0$, then the columns $\mathbf{q}_j$ of $\mathbf{Q}_c$ with indices $j = n+1, \ldots, n+m-s$ are linearly dependent with respect to the first n columns in $\mathbf{Q}_c$. This dependence could be represented by:

$$\left[\mathbf{q}_{n+1} \ \cdots \ \mathbf{q}_{n+m-s}\right] = \left[\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_{n-1} \ \mathbf{q}_n\right] \mathbf{P} \tag{4}$$

where the vectors $\mathbf{q}_i$, $i=[1, (n+m-s)]$ represent columns of $\mathbf{Q}_c$ and the matrix $\mathbf{P} = \{p_{ij}\}$ is the $n \times (m-s)$ coefficient matrix. Let the last s rows in $\mathbf{P}$ be denoted by the $[s \times (m-s)]$ matrix $\mathbf{C}$, i.e.:

$$\mathbf{P} = \left[\begin{array}{c} \mathbf{R} \\ ---- \\ \mathbf{C} \end{array} \right] \begin{array}{l} \} \ n-s \\ \\ \} \ s \end{array} \tag{5}$$

Using Eqs.(4) and (5), the columns $\mathbf{q}_{n+j}$, $j = [1,m-s]$, could be written as:

$$\mathbf{q}_{n+j} = \sum_{i=1}^{n-s} \mathbf{q}_i r_{ij} + \sum_{i=1}^{s} \mathbf{q}_{n-s+i} c_{ij} \tag{6}$$

We are now ready to formulate the main result, necessary and sufficient conditions for the set of controllability indices to be invariant under any ordering of inputs.

Theorem

The set of controllability indices $\mu = \{\mu_1, \ldots, \mu_m\}$ of a given pair $\{\mathbf{A},\mathbf{B}\}$ is invariant under any ordering of columns of $\mathbf{B}$ if and only if the following two conditions are satisfied:

1. The indices μ_i are generic, i.e. they satisfy Eq.(1), and
2. All elements c_{ij} of the $[s \times (m-s)]$ matrix $\mathbf{C} = \{ c_{ij} \}$ defined by Eqs.(4) and (5) are different from zero.

The Theorem above implies specifically that if some $c_{ij} = 0$, $i=[1, s]$, $j=[1, m\text{-}s]$, then the interchange of columns $\mathbf{b}_i$ and $\mathbf{b}_{j+s}$ would yield a non-generic set of controllability indices. Then, in the case that all other c_{ij} are different from zero, in the obtained set μ the elements μ_s and μ_{s+1} would have the values:

$$\mu_s = \left[\frac{n}{m}\right], \quad \text{and} \quad \mu_{s+1} = \left[\frac{n}{m}\right] + 1 \tag{7}$$

not *vice versa.*

If more than one $c_{ij} = 0$, and if the columns $\mathbf{b}_i$ and $\mathbf{b}_{j+s}$ are interchanged, then the controllability indices of the rearranged pair $\{\mathbf{A},\mathbf{B}\}$ are not necessarily given by Eq.(7), but they will always be *non*-generic, i.e. different from Eq.(1). Obviously, if $s = 0$, i.e. if m divides n without reminder, the matrix $\mathbf{C}$ does not exist (it is empty), and then, if the set μ is generic, it is invariant under any ordering of the columns of $\mathbf{B}$.

Proof.

Consider first the special case when $c_{ij} = 0$ for $i=s$ and $j=1$. Then, from Eq.(6) and taking into account that $c_{s1} = 0$, the column $\mathbf{q}_{n+1}$ becomes:

$$\mathbf{q}_{n+1} = \sum_{i=1}^{n-s} \mathbf{q}_i r_{i1} + \sum_{i=1}^{s-1} \mathbf{q}_{n-s+i} c_{i1} \tag{8}$$

Thus, the column $\mathbf{q}_{n+1}$, i.e. the first linearly dependent column of $\mathbf{Q}_c$, is dependent only on the first $n\text{-}1$ columns of $\mathbf{Q}_c$. Therefore, if columns $\mathbf{b}_s$ and $\mathbf{b}_{s+1}$ of the input matrix $\mathbf{B}$ are interchanged, then the only difference in the rearranged controllability matrix with respect to $\mathbf{Q}_c$, given in Eq.(3), would be that the columns:

$$\mathbf{q}_{s+mk} \quad \text{and} \quad \mathbf{q}_{s+mk+1} \quad \text{for} \quad k = [0, n-m] \tag{9}$$

are interchanged. Then, checking linear dependence of columns from the left in the rearranged controllability matrix, one would encounter column $\mathbf{q}_{n+1}$ at the n^{th} location which, according to Eq.(8), is linearly *dependent* with respect to the first $n\text{-}1$ columns in $\mathbf{Q}_c$, or the rearranged $\mathbf{Q}_c$. At that moment only $n\text{-}1$ columns have been selected. The last, n^{th} linearly independent column, would be encountered in the modified controllability matrix at location $n+1$, where the linearly independent column $\mathbf{q}_n$ from the original $\mathbf{Q}_c$ is now located. Thus, the values of μ_s and μ_{s+1} would become:

$$\mu_s = \left[\frac{n}{m}\right], \quad \text{and} \quad \mu_{s+1} = \left[\frac{n}{m}\right] + 1 \tag{10}$$

leading to a non-generic set of controllability indices.

Assume now that $c_{ij} = 0$, with $i < s$ and $j > 1$. We will show that this case can be reduced to the previous special case of $i=s$ and $j=1$. To this end, consider the sequence of three rearrangements of columns of $\mathbf{B}$; namely,

1. interchange columns $\mathbf{b}_i$ and $\mathbf{b}_s$
2. interchange columns $\mathbf{b}_{s+j}$ and $\mathbf{b}_{s+1}$
3. interchange columns $\mathbf{b}_{s+1}$ and $\mathbf{b}_s$

Considering Eqs.(3) and (6), it is easy to verify that rearrangements (1) and (2) do not produce any modification in the values of controllability indices, while rearrangement (3) leads to a non-generic set given by Eq.(7) or Eq.(10).

Let us assign to each element c_{ij}, $i=[1,s]$, $j=[1,m-s]$, of the matrix $\mathbf{C} = \{ c_{ij} \}$ a pair of indices $\{i,s+j\}$ of the input matrix $\mathbf{B} = [\ \mathbf{b}_1\ \ \mathbf{b}_2\ \dots\ \mathbf{b}_m\ ]$. To visualize this assignment, let us write:

$$\mathbf{C} = \{ c_{ij} \} \ \Rightarrow \ \begin{bmatrix} \{1,s+1\} & \{1,s+2\} & \cdots & \{1,m\} \\ \{2,s+1\} & \{2,s+2\} & \cdots & \{2,m\} \\ & \cdots & \{i,s+j\} & \\ \{s,s+1\} & \{s,s+2\} & \cdots & \{s,m\} \end{bmatrix} \tag{11}$$

Thus, if an element $c_{ij} = 0$, then, in the case that the controllability indices of the pair $\{\mathbf{A},\mathbf{B}\}$ are generic, the controllability indices of the pair $\{\mathbf{A},\mathbf{B}_{i,\,s+j}\}$, where $\mathbf{B}_{i,\,s+j}$ is obtained from $\mathbf{B}$ by interchanging columns $\mathbf{b}_i$ and $\mathbf{b}_{s+j}$, would *not* be generic.

Example

In order to visualize these relationships, the following example is used for illustration. Assume that $n=13$ and $m=5$, leading to $s=3$ and $m\text{-}s=2$. Columns of the matrix $\mathbf{Q}_c$ and elements of the row $\mathbf{r}$ are:

$$\begin{array}{rl} \mathbf{Q}_c = & [\quad\quad \mathbf{B} \quad\quad \mid \quad\quad \mathbf{A\,B} \quad\quad \mid \quad\quad \mathbf{A}^2\,\mathbf{B} \quad\quad \mid \ \cdots\ ] \\ = & [\ \mathbf{b}_1\ \mathbf{b}_2\ \mathbf{b}_3\ \mathbf{b}_4\ \mathbf{b}_5 \mid \mathbf{A\,b}_1\ \mathbf{A\,b}_2\ \mathbf{A\,b}_3\ \mathbf{A\,b}_4\ \mathbf{A\,b}_5 \mid \quad \cdots \quad \mid\ \cdots\ ] \\ \mathbf{r} = & [\ r_1\ r_2\ r_3\ r_4\ r_5 \mid r_6\ \ r_7\ \ r_8\ \ r_9\ \ r_{10} \mid r_{11}\ r_{12}\ r_{13}\ r_{14}\ r_{15} \mid\ \cdots\ ] \end{array} \tag{12}$$

In the generic case

$$\mathbf{r} = [\ 1\ 1\ 1\ 1\ 1 \mid 1\ 1\ 1\ 1\ 1 \mid 1\ 1\ 1\ 0\ 0 \mid\ \cdots\ ]$$

i.e. $r_j = 1$ for $j = [1,\ 13]$ and $r_j = 0$ for $j > 13$, leading to

$$\boldsymbol{\mu} = \{\ 3\ \ 3\ \ 3\ \ 2\ \ 2\ \}$$

If we interchange $\mathbf{b}_i$ with $\mathbf{b}_j$, for any i and $j=[1,\ 3]$, or if we interchange $\mathbf{b}_4$ with $\mathbf{b}_5$, then from Eq.(12) it is clear that the set $\boldsymbol{\mu}$ remains unchanged. However, if $\mathbf{b}_3$ is interchanged with $\mathbf{b}_4$, then the set $\boldsymbol{\mu}$ would become:

$$\boldsymbol{\mu} = \{\ 3\ \ 3\ \ 2\ \ 3\ \ 2\ \}$$

only if the linearly dependent column $\mathbf{A}^2\,\mathbf{b}_4$ is dependent only on the first $n\text{-}1=12$ columns, i.e. it is not dependent on the column $\mathbf{A}^2\,\mathbf{b}_3$. Clearly, this situation occurs only if the coefficient $c_{31} = 0$, which could be verified by considering Eqs.(5), (6) and (8). The same arguments apply even if the given pair {**A**,**B**} is not controllable.

Numerical Example

As a counter example to the statements given in references [1] and [2], and in order to clarify the above discussion even further, the following example is treated. Consider a pair {**A**,**B**} with $n=8$, $m=5$, leading to $s=3$ and $m\text{-}s=2$, where:

$$\mathbf{A} = diag\{\ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \}$$

and

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

It may be easily shown that the controllability indices of {**A**,**B**} are:

$$\boldsymbol{\mu} = \{\ 2\ \ 2\ \ 2\ \ 1\ \ 1\ \} \tag{13}$$

The $s \times m\text{-}s$ matrix **C** defined by Eqs.(4) and (5) and the associated pairs $\{i,s+j\}$ of indices of the columns of **B** become:

$$\mathbf{C} = \begin{bmatrix} 1.0 & 11.0 \\ 0.0 & 3.0 \\ -0.5 & 0.0 \end{bmatrix} \Rightarrow \begin{bmatrix} \{1,4\} & \{1,5\} \\ \{2,4\} & \{2,5\} \\ \{3,4\} & \{3,5\} \end{bmatrix}$$

Since the elements c_{32} and c_{21} are equal to zero, the controllability indices of the pairs {**A**,**B**$_{35}$} and {**A**,**B**$_{24}$}, where:

$$\mathbf{B}_{35} = [\ \mathbf{b}_1\ \ \mathbf{b}_2\ \ \mathbf{b}_5\ \ \mathbf{b}_4\ \ \mathbf{b}_3\ ] \quad \text{and} \quad \mathbf{B}_{24} = [\ \mathbf{b}_1\ \ \mathbf{b}_4\ \ \mathbf{b}_3\ \ \mathbf{b}_2\ \ \mathbf{b}_5\ ] \tag{14}$$

in both cases yield:

$$\boldsymbol{\mu} = \{\ 2\ \ 2\ \ 1\ \ 2\ \ 1\ \}$$

while all other interchanges of columns in **B** give the generic set of controllability indices in Eq.(13).

References

[1] C. T. Chen, *Linear System Theory and Design*, Holt, Rinehart and Winston, New York, NY, (1984).

[2] T. Kailath, *Linear Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, (1980).

ALTERNATING CONVEX PROJECTION METHOD FOR COVARIANCE CONTROL DESIGN

Karolos M. Grigoriadis and Robert E. Skelton
Space Systems Control Laboratory, Purdue University, W. Lafayette, IN 47907

Abstract

An alternating convex projection methodology is suggested to solve the covariance control design problem, i.e., to find a closed loop covariance matrix which is assignable to the closed loop system and which satisfies desired output performance objectives. Two aspects of the problem are addressed: In the covariance feasibility problem we look for a covariance matrix to satisfy assignability and output performance constraints. In the covariance optimization problem we find an assignable covariance matrix which satisfies output performance constraints and is as close as possible to a given desired one. Expressions for the required convex projections on the assignability and the performance constraint sets are derived.

1. Introduction

Many engineering control design problems have performance requirements expressed as variance constraints on each output of the system. The theory of covariance control was originally developed to provide a direct approach for achieving these kinds of multiobjective performance goals [1-3]. The multiobjective nature of the problem is embedded in the closed loop state covariance matrix and covariance control theory provides the necessary and sufficient conditions for such a matrix to be "assignable". In addition, this theory provides explicitly the set of all controllers which assign a particular matrix as a closed loop system covariance. Covariance control theory can also be interpreted as a parametrization of all stabilizing controllers of fixed order [4].

The difficulty in using the covariance control theory for control design is that there is no systematic procedure to find a desirable covariance matrix which is also assignable to the closed loop system. This problem is addressed in this paper. We show that the set of all assignable plant state covariances which simultaneously satisfy a certain desired set of performance objectives is a convex set. Projection type algorithms are proposed for the numerical solution.

The covariance design problems treated herein are two:

1) The covariance feasibility problem: In this problem we find an assignable covariance which satisfies output performance constraints.
2) The covariance optimization problem: In this problem we find an assignable covariance which satisfies output performance constraints and is as close as possible to a given desired (but non-assignable) one.

We will show that these two problems are convex. Hence efficient numerical techniques can be applied to find a solution. In this paper we examine simple and effective numerical techniques based on alternating convex projection methods [6-8]. These methods require analytic expressions for the projection on each individual constraint set, hence expressions for the projections on the assignability and performance constraint sets are derived. A simple example illustrates the procedure. Our development and results are for the continuous case but a similar approach can be followed for the discrete time case. Proofs have been ommited and will appear in a full version of the paper.

2. Preliminaries

Consider a linear, time-invariant, continuous time dynamical system with the following state space representation

$$\dot{x}_p = A_p x_p + B_p u + D_p w_p\,, \quad y = C_p x_p\,, \quad z = M_p x_p + v \tag{2.1}$$

where $x_p(t) \in \mathbb{R}^{n_x}$ is the plant state vector, $y(t) \in \mathbb{R}^{n_y}$ is the vector of outputs whose performances are of interest, and $z(t) \in \mathbb{R}^{n_z}$ is the vector of noisy measurements. We assume that (A_p, D_p) is controllable, (A_p, B_p) is stabilizable and (A_p, M_p) is a detectable pair. In a stochastic interpretation of the system (2.1), $w_p(t) \in \mathbb{R}^{n_w}$ and $v(t) \in \mathbb{R}^{n_z}$ are zero mean white noise processes with intensities $W_p > 0$ and $V > 0$, respectively. We seek a dynamic controller of order n_c with state space representation

$$\dot{x}_c = A_c x_c + B_c z\,, \quad u = C_c x_c + D_c z \tag{2.2}$$

where $x_c(t) \in \mathbb{R}^{n_c}$.

Combining eqns. (2.1) and (2.2), the closed loop system can be expressed in the following form

$$\dot{x} = (A+BGM)x + (D+BGJ)w \tag{2.3}$$

where

$$x \triangleq \begin{bmatrix} x_p \\ x_c \end{bmatrix}, \quad w \triangleq \begin{bmatrix} w_p \\ v \end{bmatrix} \tag{2.4}$$

and

$$A \triangleq \begin{bmatrix} A_p & 0 \\ 0 & 0 \end{bmatrix}, \quad B \triangleq \begin{bmatrix} B_p & 0 \\ 0 & I_{n_c} \end{bmatrix}, \quad M \triangleq \begin{bmatrix} M_p & 0 \\ 0 & I_{n_c} \end{bmatrix}, \quad D \triangleq \begin{bmatrix} D_p & 0 \\ 0 & 0 \end{bmatrix}, \quad J \triangleq \begin{bmatrix} 0 & I_{n_c} \\ 0 & 0 \end{bmatrix}, \quad G \triangleq \begin{bmatrix} D_c & C_c \\ B_c & A_c \end{bmatrix}. \tag{2.5}$$

The closed loop state covariance

$$X \triangleq \lim_{t \to \infty} \mathcal{E}\,\{x(t)x^T(t)\} \tag{2.6}$$

satisfies the following Lyapunov equation

$$(A+BGM)X + X(A+BGM)^T + (D+BGJ)W(D+BGJ)^T = 0 \tag{2.7}$$

where the noise covariance matrix W and the state covariance matrix X are decomposed as follows

$$W \triangleq \begin{bmatrix} W_p & 0 \\ 0 & V \end{bmatrix}, \quad X = \begin{bmatrix} X_p & X_{pc} \\ X_{pc}^T & X_c \end{bmatrix}. \tag{2.8}$$

Lyapunov stability theory guarantees that the closed loop system is asymptotically stable (i.e. every eigenvalue of A+BGM has a negative real part) if and only if $X > 0$. Hence we are interested only in positive definite state covariances which satisfy eq. (2.7).

Definition 2.1

A closed loop state covariance $X > 0$ is *assignable* to the closed loop system if X satisfy eq. (2.7) for some G.

It turns out that for our purposes it is more tractable to work with the plant state covariance X_p. Hence we need the following definition.

Definition 2.2

A plant state covariance $X_p > 0$ is *assignable* if there exist X_c and X_{pc} such that $X > 0$ is assignable.

Note that $X_p > 0$ and assignability guarantees stabilizability of the system. We will need the following result [5].

Theorem 2.1

The set of all assignable $X_p > 0$ is parameterized by

$$(I - B_p B_p^+)(A_p X_p + X_p A_p^T + D_p W_p D_p^T)(I - B_p B_p^+) = 0\,, \quad X_p \geq P \tag{2.9a}$$

$$A_p P + P A_p^T - P M_p^T V^{-1} M_p P + D_p W_p D_p^T = 0\,, \quad W_p > 0\,, \quad V > 0\,. \tag{2.9b}$$

Covariance control theory provides necessary and sufficient conditions for assignability of a given state covariance X and gives explicit formulas for all controllers which assign this X to the closed loop system. Hence, covariance control theory provides a state-space parametrization of all fixed order stabilizing controllers [4].

A very important fact about the state covariance X is that many closed loop performance objectives can be expressed in terms of the plant parameters $(A_p, B_p, D_p, C_p, M_p)$ and X without involving the unknown controller parameters (A_c, B_c, C_c, D_c). Therefore, covariance control theory allows the control design problem to be posed in terms of an assignable state covariance $X > 0$. As long as a desired $X > 0$ is found, covariance control theory provides directly the analytic expressions for a controller which assigns this X to the closed loop system. The expressions for the controllers may be found in [4-5]. Our focus here is to find an assignable covariance X to be used in the controller design.

In the next section certain objectives for the closed loop system will be expressed geometrically, as requirements that the plant state covariance X_p belongs to specified closed and convex sets in the space of $n_x \times n_x$ real symmetric matrices.

3. Convex Constraint Formulation of the Covariance Design Problem

We refer to the problem of constructing an assignable covariance X as the *Covariance Design Problem.* Consider the $n_x(n_x+1)/2$ dimensional vector space $\mathcal{S}_{n_x}$ of $n_x \times n_x$ real symmetric matrices. We will now characterize certain properties of the plant covariance matrix X_p as convex and closed sets in $\mathcal{S}_{n_x}$.

Assignability Constraints on $\mathcal{S}_{n_x}$

To interpret the necessary and sufficient conditions for assignability of a plant state covariance matrix in a geometric setting, we define the following subsets of $\mathcal{S}_{n_x}$

$$\mathcal{A} \triangleq \{X_p \in \mathcal{S}_{n_x} \,/\, (I-B_pB_p^+)(A_pX_p+X_pA_p^T+D_pW_pD_p^T)(I-B_pB_p^+) = 0\} \tag{3.1}$$

and

$$\mathcal{P} \triangleq \{X_p \in \mathcal{S}_{n_x} \,/\, X_p \geq P\} \tag{3.2}$$

where P the positive definite solution of the following Riccati equation

$$A_pP + PA_p^T - PM_p^TV^{-1}M_pP + D_pW_pD_p^T = 0\,. \tag{3.3}$$

Now, the plant state covariance assignability conditions of Theorem 2.1 can be expressed as follows.

Proposition 3.1

$X_p \in \mathcal{S}_{n_x}$ *is an assignable plant state covariance if and only if*

$$X_p \in \mathcal{A} \cap \mathcal{P}\,. \tag{3.4}$$

Note that for the state feedback case the constraint set (3.2) reduces to

$$\mathcal{P} \triangleq \{X_p \in \mathcal{S}_{n_x} \,/\, X_p > 0\} \tag{3.5}$$

and since we will require the constraint sets to be closed, we replace $\mathcal{P}$ in (3.5) by

$$\mathcal{P}_\varepsilon \triangleq \{X_p \in \mathcal{S}_{n_x} \,/\, X_p \geq \varepsilon I_{n_x}\} \tag{3.6}$$

for an arbitrarily small $\varepsilon > 0$.

The following result provides a very important geometric property of the set of all assignable plant state covariances.

Proposition 3.2

The set of all assignable plant state covariances is a closed and convex subset of $\mathcal{S}_{n_x}$.

Performance Constraints on $\mathcal{S}_{n_x}$

Variance performance requirements on the system outputs correspond to inequality constraints on the diagonal elements of the output covariance matrix

$$Y = C_pX_pC_p^T\,. \tag{3.7}$$

Suppose that the desirable output variance constraints (OVC) are

$$[C_pX_pC_p^T]_{ii} \leq \sigma_i \qquad i = 1, \ldots, n_y \tag{3.8}$$

where for simplicity we assume that C_p is a full row rank matrix. Equivalently, (3.8) can be expressed as

$$c_{pi}X_pc_{pi}^T \leq \sigma_i \qquad i = 1, \ldots, n_y \tag{3.9}$$

where c_{pi}, $i = 1, \ldots, n_y$ the rows of C_p. Let's define the following subsets of $\mathcal{S}_{n_x}$

$$\mathcal{V}_i = \{X_p \in \mathcal{S}_{n_x} \,/\, c_{pi}X_pc_{pi}^T \leq \sigma_i\} \qquad i = 1, \ldots, n_y\,. \tag{3.10}$$

Then the requirement that the system satisfy the ith variance constraint can be expressed geometrically as

$$X_p \in \mathcal{V}_i \tag{3.11}$$

The following result can be easily proved using the definition of convexity.

Proposition 3.3

$\mathcal{V}_i$ as defined by (3.10) *is a closed and convex subset of $\mathcal{S}_{n_x}$.*

The subset of $\mathcal{S}_{n_x}$ where all the desired variance constraints are satisfied is

$$\mathcal{V} = \mathcal{V}_1 \cap \ldots \cap \mathcal{V}_{n_y} . \tag{3.12}$$

The following is an obvious consequence of Proposition 3.3.

Proposition 3.4

$\mathcal{V}$ is a closed and convex subset of $\mathcal{S}_{n_x}$.

Next, we consider a constraint on the weighted closed loop system output cost defined as

$$J_y \triangleq \lim_{t \to \infty} \mathcal{E}\,\{y^T(t) Q y(t)\} \tag{3.13}$$

where $Q \geq 0$ is a constant weighting matrix. It can be easily verified that

$$J_y = \mathrm{tr}(X_p C_p^T Q C_p) . \tag{3.14}$$

Now, suppose that the desired output cost constraint is

$$J_y \leq \gamma \tag{3.15}$$

where $\gamma > 0$. Then by defining the set

$$\mathcal{C} = \{X_p \in \mathcal{S}_{n_x} \;/\; \mathrm{tr}(X_p C_p^T Q C_p) \leq \gamma\} \tag{3.16}$$

the desired constraint can be written as

$$X_p \in \mathcal{C} . \tag{3.17}$$

The following result is easy to verify

Proposition 3.5

$\mathcal{C}$ is a closed and convex subset of $\mathcal{S}_{n_x}$.

From the discussion above we conclude that the covariance control design problem subject to OVC and output cost constraints of the form (3.8) and (3.15) respectively can be posed as a feasibility problem in the space $\mathcal{S}_{n_x}$ of $n_x \times n_x$ real symmetric matrices, as follows:

Covariance Feasibility Problem (CFP)

Find $X_p^* \in \mathcal{S}_{n_x}$ such that

$$X_p^* \in \mathcal{A} \cap \mathcal{P} \cap \mathcal{V}_1 \cap \ldots \cap \mathcal{V}_{n_y} \cap \mathcal{C} . \tag{3.18}$$

Note that the corresponding set in (3.18) is closed and convex as an intersection of a finite number of closed and convex sets.

The covariance control design problem can alternatively be posed as an optimization problem in $\mathcal{S}_{n_x}$. Suppose that $\bar{X}_p > 0$ denotes a "desirable" plant state covariance, constructed to represent some desired system performance. Note that $\bar{X}_p$ might not be assignable. Hence, we are interested to find an assignable covariance X_p^* which satisfy the desired performance objectives (3.8) and (3.15) and is as close as possible to $\bar{X}_p$.

Covariance Optimization Problem (COP)

Given $\bar{X}_p \in \mathcal{S}_{n_x}$ find $X_p^* \in \mathcal{S}_{n_x}$ to solve

$$\min \| \bar{X}_p - X_p^* \| \quad \text{such that} \quad X_p^* \in \mathcal{A} \cap \mathcal{P} \cap \mathcal{V}_1 \cap \ldots \cap \mathcal{V}_{n_y} \cap \mathcal{C} . \tag{3.19}$$

For mathematical convenience the matrix norm we will use in the above optimization problem is the Frobenious norm $\| X \|_F = [\mathrm{tr}\,(X^2)]^{1/2}$. Note that since the underlying set is closed and convex and the minimized function is also convex, the covariance optimization problem is a nonlinear convex programming problem.

Next a method of alternating projections is used to solve both CFP and COP.

4. The Alternating Convex Projections Method

Consider a finite dimensional Hilbert space $\mathcal{S}$ and let $\|\cdot\|$ be the norm induced by the inner product $<\cdot,\cdot>$. We are interested to solve the following problems:

Feasibility Problem

Given closed and convex sets $C_1, \ldots, C_n$ in $\mathcal{S}$ such that $C_1 \cap C_2 \cap \ldots \cap C_n \neq$ emplty, find vector $x \in \mathcal{S}$ such that

$$x \in C_1 \cap C_2 \cap \ldots \cap C_n . \tag{4.1}$$

Optimization Problem

Given closed and convex sets $C_1, \ldots, C_n$ in $\mathcal{S}$ such that $C_1 \cap C_2 \cap \ldots \cap C_n \neq \emptyset$ and $x_o \in \mathcal{S}$, find vector $x \in \mathcal{S}$ to solve

$$\min \| x - x_o \| \quad \text{such that} \quad x \in C_1 \cap C_2 \cap \ldots \cap C_n . \tag{4.2}$$

Note that the Optimization Problem (4.2) has a unique solution according to the following well known result (e.g., see [10]).

Theorem 4.1

Let $\mathcal{K} \neq$ emplty be a closed and convex set in $\mathcal{S}$ and $x_o \in \mathcal{S}$. Then there exist a unique $x^* \in \mathcal{K}$ which is closest to x_o, i.e., x^* satisfies

$$\| x_o - x^* \| \leq \| x_o - \hat{x} \| \qquad \forall \hat{x} \in \mathcal{K} . \tag{4.3}$$

We will use the following definition.

Definition 4.1

We will call $x^* \in \mathcal{K}$ which solves (4.3) the *projection* of x on $\mathcal{K}$ and we will denote this by $x^* \triangleq P_{\mathcal{K}}(x)$.

A simple algorithm to solve the Feasibility Problem (4.1) and the Optimization Problem (4.2) is provided by the method of alternating convex projections [6-8]. This method consists of cyclic successive projections into each individual convex set C_i, $i = 1, \ldots, n$. The standard method [6] solves problem (4.1) and a modification of the method [7] can solve problem (4.2). This method has been applied successfully in statistics, signal restoration, image reconstruction problems and other fields of data analysis. A basic feature of the convex projections method is that an expression for the projection $P_{C_i}(x)$ of x on each individual convex set C_i is required, hence the corresponding convex sets should be analytically tractable enough so that such an expression can be derived.

The convex projections method can be described as follows:

Alternating Projection Algorithm for the Feasibility Problem

Let $x_o \in \mathcal{S}$ and define:

$$\begin{aligned}
&\left. x_1 = P_{C_1} x_o , \quad x_2 = P_{C_2} x_1 , \ldots , x_n = P_{C_n} x_{n-1} \right\} \text{1}^{\text{st}} \text{ cycle} \\
&\left. x_{n+1} = P_{C_1} x_n , \; x_{n+2} = P_{C_2} x_{n+1} , \ldots , x_{2n} = P_{C_n} x_{2n-1} \right\} \text{2}^{\text{nd}} \text{ cycle} \\
&\quad \vdots
\end{aligned} \tag{4.4}$$

Hence, the sequence of vectors $\{x_k\}_{k=0}^{\infty}$ is formed by successive projections in a cyclic manner on the convex sets C_i, $i = 1, \ldots, n$. A sketch of the method for $n = 2$ is given in Figure 1. The following result (given in [6]) points out the global convergence property of the sequence $\{x_k\}_{k=0}^{\infty}$.

Theorem 4.2

The sequence $\{x_k\}_{k=0}^{\infty}$ generated by (4.4) converges to a vector $x \in C_1 \cap C_2 \cap \ldots \cap C_n$ for any initial vector $x_o \in \mathcal{S}$.

Hence the algorithm (4.4) solves the Feasibility Problem (4.1).

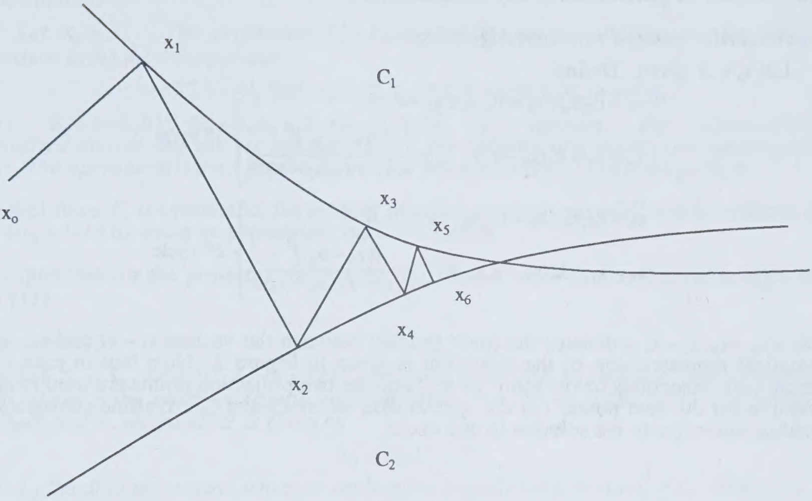


Figure 1

A modification of the algorithm (4.4) can be used to solve the Optimization Problem (4.2).

Alternating Projection Algorithm for the Optimization Problem

Let $x_o \in \mathcal{S}$ and define

$$\left.\begin{array}{lcl} x_1 = P_{C_1}x_o & , & z_1 = x_1 - x_o \\ x_2 = P_{C_2}x_1 & , & z_2 = x_2 - x_1 \\ \vdots & & \vdots \\ x_n = P_{C_n}x_{n-1} & , & z_n = x_n - x_{n-1} \end{array}\right\} 1^{st}\text{ cycle}$$

$$\left.\begin{array}{lcl} x_{n+1} = P_{C_1}(x_n - z_1) & , & z_{n+1} = x_{n+1} - (x_n - z_1) \\ x_{n+2} = P_{C_2}(x_{n+1} - z_2) & , & z_{n+2} = x_{n+2} - (x_{n+1} - z_2) \\ \vdots & & \vdots \\ x_{2n} = P_{C_n}(x_{2n-1} - z_n) & , & z_{2n} = x_{2n} - (x_{2n-1} - z_n) \end{array}\right\} 2^{nd}\text{ cycle} \qquad (4.5)$$

$$\begin{array}{lcl} x_{2n+1} = P_{C_1}(x_{2n} - z_{n+1}) & & z_{2n+2} = x_{2n+2} - (x_{2n+1} - z_{n+1}) \\ \vdots & & \vdots \end{array}$$

Note that in this modified projection algorithm in each step j, the increment z_{j-n} is removed before projecting on the corresponding convex set.

The next result guarantees that the algorithm (4.5) converges to a solution of the optimization problem (4.2).

Theorem 4.3 [7]

The sequence $\{x_k\}_{k=o}^{\infty}$ generated by (4.5) converges to $x^* \in C_1 \cap C_2 \cap \ldots \cap C_n$ where x^* satisfies

$$\| x_o - x^* \| \leq \| x_o - \hat{x} \| \quad \forall\ \hat{x} \in C_1 \cap C_2 \cap \ldots \cap C_n . \qquad (4.6)$$

An important feature of the feasibility and optimization alternating projection algorithms (4.4) and (4.5) is that they can be realized very simply and usually the amount of calculations in one iteration is very small. However, a serious drawback of the method may be its slow convergence. Possible ways to accelerate the rate of convergence has been examined in [6]. A very effective method which we will call the *Directional Alternating Projection Method* will be examined next. For simplicity we will examine the case $n = 2$ but the method can be generalized to any finite number of closed and convex sets.

Directional Alternating Projection Algorithm

Let $x_o \in \mathcal{S}$ given. Define

$$\begin{aligned}
&\left.\begin{aligned} & x_1 = P_{C_1}x_o\,,\ x_2 = P_{C_2}x_1\,,\ x_3 = P_{C_1}x_2 \\ & x_4 = x_1 + \lambda_1(x_3 - x_1)\,,\ \lambda_1 = \frac{\| x_1 - x_2 \|^2}{< x_1 - x_3,\, x_1 - x_2 >} \end{aligned}\right\} \ 1^{\text{st}} \text{ cycle} \\
&\left.\begin{aligned} & x_5 = P_{C_1}x_4\,,\ x_6 = P_{C_2}x_5\,,\ x_7 = P_{C_1}x_6 \\ & x_8 = x_7 + \lambda_2(x_7 - x_5)\,,\ \lambda_2 = \frac{\| x_5 - x_6 \|^2}{< x_5 - x_7,\, x_5 - x_6 >} \end{aligned}\right\} \ 2^{\text{nd}} \text{ cycle} \\
&\quad \vdots
\end{aligned} \tag{4.7}$$

where $< x_1 - x_3, x_1 - x_2 >$ denotes the inner product between the vectors $x_1 - x_3$ and $x_1 - x_2$. A schematical representation of the algorithm is given in Figure 2. Note that in each cycle, there are two projections on the same set (C_1) and the two projection points are used to derive a direction for the next point. For the special case where C_1 and C_2 are affine subspaces, the algorithm converges to the solution in one cycle.

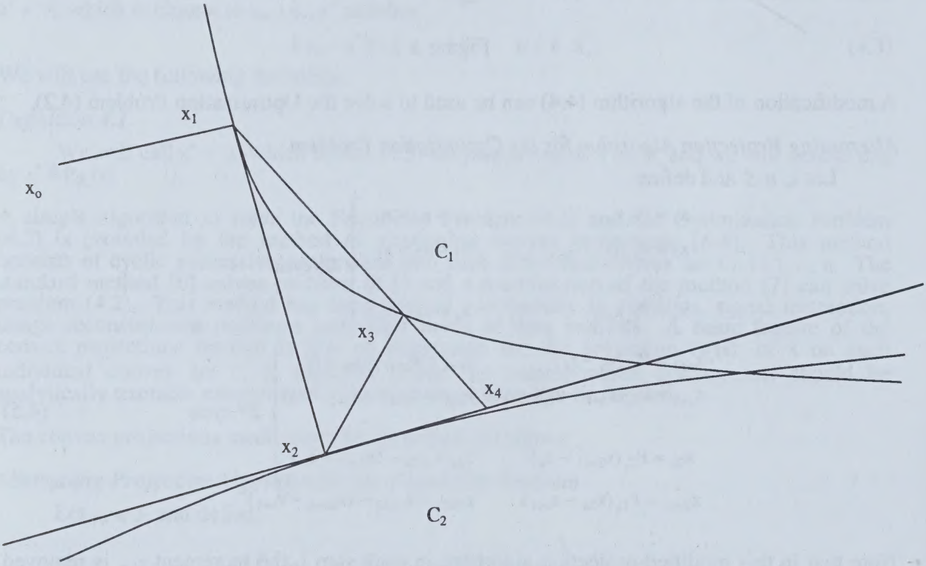


Figure 2

Convergence of the method to a solution of the feasibility problem (4.1) is proved in [6].

5. Alternating Convex Projection Method for Covariance Design

In this section we will use the alternating convex projection method described in section 4, to solve the Covariance Feasibility Problem (CFP) and the Covariance Optimization Problem (COP) described in (3.16) and (3.17), respectively.

Lets consider the vector space $\mathcal{S}_{n_x}$ of $n_x \times n_x$ real symmetric matrices equipped with the inner product and the corresponding induced norm (Frobenious norm)

$$< X, Y > = [\mathrm{tr}\,(XY)]^{1/2}\,,\quad \| X \| = [\mathrm{tr}\,(X^2)]^{1/2} = \left[\sum_{i,j=1}^{n_x} X_{ij}^2 \right]^{1/2}\,,\qquad X, Y \in \mathcal{S}_{n_x}\,. \tag{5.1}$$

To be able to use the alternating projection algorithms of Section 4 we need to derive expressions for the projection operators on each of the assignability and performance constrained sets $\mathcal{A}$, $\mathcal{P}$, $\mathcal{V}_i$, $i = 1, \ldots, n_y$ and $\mathcal{C}$ defined in Section 3. This is our next task.

Proposition 5.1

Let $X_p \in \mathcal{S}_{n_x}$. *The projection* $X_p^* = P_{\mathcal{A}}(X_p)$ *of* X_p *on* $\mathcal{A}$ *is given by the following expression using the vec operator*

$$\operatorname{vec} X_p^* = -[E_p \otimes (E_p A_p) + (E_p A_p) \otimes E_p]^+ \operatorname{vec}(E_p Q_p E_p) + \operatorname{vec} X_p \tag{5.2}$$

where $E_p = I - B_p B_p^+$, $Q_p = A_p X_p + X_p A_p^T + D_p W_p D_p^T$, $[\cdot]^+$ *denotes the Moore-Penrose generalized inverse and the vec operator stacks the columns of a matrix one underneath the other. The operator* $\otimes$ *is used for the Kronecker product* $M \otimes N = L$, *where* $L_{ij} = M_{ij} N$.

Note that since X_p^* is symmetric, the number of unknown elements of X_p^* can be reduced from n_x^2 to $n(n_x + 1)/2$ by using an appropriate injection matrix.

The expression for the projection of any $X_p \in \mathcal{S}_{n_x}$ on the set $\mathcal{P}$ can be derived using a result from [11].

Proposition 5.2

Let $X_p \in \mathcal{S}_{n_x}$ *and let* $X_p - P = ULU^T$ *be the eigenvalue-eigenvector decomposition of* $X_p - P$ *where* P *the positive definite solution of the Riccati equation* (3.3). *The projection* $X_p^* = P_{\mathcal{P}}(X_p)$ *of* X_p *on the set* $\mathcal{P}$ *is given by*

$$X_p^* = UL_+U^T + P \tag{5.3}$$

where L_+ *the diagonal matrix when we replace the negative eigenvalues of* $X_p - P$ *by zero.*

Hence this projection requires an eigenvalue-eigenvector decomposition of an $n_x \times n_x$ symmetric matrix $X_p - P$.

Next we will provide the expression for the projection on the set of ith variance constraint $\mathcal{V}_i$.

Proposition 5.3

Let $X_p \in \mathcal{S}_{n_x}$. *The projection* $X_p^* = P_{\mathcal{V}_i}(X_p)$ *of* X_p *on* $\mathcal{V}_i$ *is given by :*

$$X_p^* = (\sigma_i^* - c_{pi} X_p c_{pi}^T) c_{pi}^+ c_{pi}^{+T} + X_p \tag{5.4}$$

where

$$\sigma_i^* = \begin{cases} \sigma_i & \text{if } (C_p X_p C_p^T)_{ii} > \sigma_i \\ (C_p X_p C_p^T)_{ii} & \text{if } (C_p X_p C_p^T)_{ii} \le \sigma_i \end{cases} \tag{5.5}$$

and c_{pi} *is the ith row of the output matrix* C_p.

For the special case where the output variance constraints are expressed only in terms of the diagonal elements of X_p, (i.e., when the rows of C_p are obtained from the rows of the identity matrix I_{n_y}) there is a very simple expression for the projection on the set of all variance constraints $\mathcal{V}$. In this case $\mathcal{V}$ can be described as

$$\mathcal{V} = \{X_p \in \mathcal{S}_{n_x} \mid (X_p)_{i_1 i_1} \le \sigma_{i_1}, \ldots, (X_p)_{i_{n_y} i_{n_y}} \le \sigma_{i_{n_y}},\ 1 \le i_1, \ldots, i_{n_y} \le n_x\} \tag{5.6}$$

i.e., the indices $i_1, \ldots, i_{n_y}$ correspond to the diagonal elements of X_p which are constrained by $\sigma_{i_1}, \ldots, \sigma_{i_{n_y}}$. The projection on $\mathcal{V}$ is given by the following result.

Proposition 5.4

Let $X_p \in \mathcal{S}_{n_x}$ *and let the set of variance constraints* $\mathcal{V}$ *be as in* (5.6). *Then the projection* $X_p^* = P_{\mathcal{V}}(X_p)$ *of* X_p *on* $\mathcal{V}$ *is given by the following expression*

$$(X_p^*)_{ij} = \begin{cases} (X_p)_{ij} & \text{if } i \ne j \text{ or } i = j \notin \{i_1, \ldots, i_{n_y}\} \\ \sigma_{i_k} & \text{if } i = j = i_k \in \{i_1, \ldots, i_{n_y}\} \text{ and } (X_p)_{i_k i_k} > \sigma_{i_k} \\ (X_p)_{i_k i_k} & \text{if } i = j = i_k \in \{i_1, \ldots, i_{n_y}\} \text{ and } (X_p)_{i_k i_k} \le \sigma_{i_k} \end{cases}$$

i.e., X_p^* is obtained when we replace the diagonal elements of X_p which are subject to variance constraints and they are bigger than σ_i, with their upper bound σ_i.

Next, the projection on the output cost constrained set C defined in (3.16) will be examined.

Proposition 5.5

Let $X_p \in \mathcal{S}_{n_x}$ *and* $\gamma > 0$ *given. Then the projection* $X_p^* = P_C(X_p)$ *of* X_p *on* C *is given by*

$$\mathrm{vec}\,(X_p^*) = [\mathrm{vec}\,(R)]^{+T}\{\gamma^* - [\mathrm{vec}\,(R)]^T\,\mathrm{vec}\,(X_p)\} + \mathrm{vec}\,(X_p) \tag{5.7}$$

where $R = C_p^T Q C_p$ *and*

$$\gamma^* = \begin{cases} \mathrm{tr}\,(X_p C_p^T Q C_p) & \text{if } \ \mathrm{tr}\,(X_p C_p^T Q C_p) \le \gamma \\ \gamma & \text{if } \ \mathrm{tr}\,(X_p C_p^T Q C_p) > \gamma \end{cases} \tag{5.8}$$

The above expression for the projections can be directly used to solve the Covariance Feasibility Problem (3.16) or the Covariance Optimization problem (3.17) using the alternating projection algorithms of section 4. When a desired plant state covariance X_p is found, then an assignable covariance X can be constructed by choosing X_c, X_{pc} in (2.8) such that

$$X_{pc} X_c^{-1} X_{pc}^T = X_p - P \ge 0 \tag{5.9}$$

Covariance control theory provides expressions for all controllers which assign this X to the closed loop system (see [4-5]). The free parameters in the controllers parameterization can be chosen to minimize the required control energy as in [12]. Note that the order n_c of the designed controller is

$$n_c = \mathrm{rank}\,(X_p - P) \tag{5.10}$$

where P is the solution of the Riccati equation (3.3) and X_p the resulted plant state covariance matrix which solves the CFP or the COP.

6. Numerical Example

As a simple numerical example consider the state space model of a fighter aircraft provided in [13]

$$A_p = \begin{bmatrix} -0.0226 & -36.6170 & -18.8970 & -32.0900 & 3.2509 & -0.7626 \\ 0.0001 & -1.8997 & 0.9831 & -0.0007 & -0.1708 & -0.0050 \\ 0.0123 & 11.7200 & -2.6316 & 0.0009 & -31.6040 & 22.3960 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -30.0000 & 0 \\ 0 & 0 & 0 & 0 & 0 & -30.0000 \end{bmatrix}$$

$$B_p = D_p = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 30 & 0 \\ 0 & 30 \end{bmatrix}, \quad C_p = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad W_p = 0.01$$

We assume that all states are available for feedback, i.e., $z = x_p$. For an initial starting covariance matrix we choose $X_p = 2\,I$ and output variance constraints $[\sigma_1 \ \ \sigma_2] = [0.01 \ \ 0.01]$ the directional alternating projection algorithm (4.6) provides the following assignable covariance

$$X_p = \begin{bmatrix} 2.0056 & 0.0101 & -0.0328 & -0.0567 & -0.5350 & 0.3757 \\ 0.0101 & 0.0028 & 0.0060 & -0.0039 & 0.0046 & -0.0525 \\ -0.0328 & 0.0060 & 1.3274 & -0.0000 & -0.0595 & 0.0688 \\ -0.0567 & -0.0039 & 0.0000 & 0.0100 & 0.0767 & 0.0511 \\ -0.5350 & 0.0046 & -0.0595 & 0.0767 & 2.0009 & -0.0010 \\ 0.3757 & -0.0525 & 0.0688 & 0.0511 & -0.0010 & 2.0055 \end{bmatrix}$$

The method requires 7 iterations for an error bound 10^{-6} (sum of the distances from the constrained sets). Note that the output variance constraint which corresponds to the first output y_1 (2×2 element of X_p) is not binding although the one which corresponds to y_2 (4×4 element of X_p) is binding (i.e., is reaching the allowed bound 0.01). A controller which assigns X_p to the closed loop system (setting the free parameters to minimize the required control effort [12]) is

$$G = \begin{bmatrix} -0.0103 & 26.5768 & 1.4216 & 14.0812 & 0.3636 & 0.2426 \\ 0.0402 & -10.6659 & -1.0383 & -4.9792 & 0.2426 & -0.8011 \end{bmatrix}$$

Conclusions

Covariance control theory allows the control design problem to be posed in terms of assigning the closed loop state covariance matrix X. This paper shows that the design problem in terms of X has some desirable convex properties. The corresponding assignability and performance constrained sets are convex sets and moreover expressions for the projections on these are analytically derived. This allows the use of alternating convex projection methods to solve the covariance design problem. Improved convergence rates are obtained by using directional information. The algorithm presented can solve the following problems by convex programming; a feasibility problem (covariance assignability subject to output system performance constraints) and an optimization problem (covariance assignability subject to output performance constraints such that the covariance is as close as possible to a given desired one). A 6th order numerical example indicates that the alternating projection type algorithms converge to a solution in a very few steps (7) when directional information are used.

Further research is needed to incorporate fixed or minimal controller order requirements into the problem.

References

[1] Hotz, A., and Skelton, R. E., "Covariance Control Theory," Int. J. Control, Vol. 46, No. 1, pp. 13-32, 1987.

[2] Skelton, R. E., and Ikeda, M., "Covariance Controllers for Linear Continuous-Time Systems," Int. J. Control, Vol. 49, No. 5, pp. 1773-1785, 1989.

[3] Skelton, R. E., and Iwasaki, T., "Lyapunov and Covariance Controllers," Proc. ACC, Chicago, Illinois, 1992.

[4] Yasuda, K., Skelton, R. E. and Grigoriadis, K. M., "Covariance Controllers: A New Parameterization of the Class of All Stabilizing Controllers," Submitted to Automatica.

[5] Iwasaki, T. and Skelton, R. E., "A Finite Step Design Algorithm for Continuous System Covariance Control," 30th Allerton Conf. Commun. Control Computing, Monticello, Illinois, 1992.

[6] Gubin, L. G., Polyak, B. T. and Raik, E. V., "The Method of Projections for Finding the Common Point of Convex Sets," USSR Comput. Math. and Mathematical Physics, Vol. 7, pp. 1-24, 1967.

[7] Boyle, J. P. and Dykstra, R. L., "A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Space," Lecture Notes in Statistics, Springer-Verlag, Vol. 37, pp. 28-47, 1986.

[8] Potter, L. C., "Convergence of Successive Approximations for Nonexpansive Operators," Proc. 29th Allerton Conf. Commun. Control Computing, pp. 149-158, 1991.

[9] Horn, R. A. and Johnson, C. R., Matrix Analysis, Cambridge University Press, 1985.

[10] Luenberger, D. G., Optimization by Vector Space Methods, John Wiley, & Sons, 1968.

[11] Higham, N. J., "Computing the Nearest Symmetric Positive Semidefinite Matrix," Linear Algebra and Its Appl., Vol. 103, pp. 103-118, 1988.

[12] Skelton, R. E., Xu, J. H. and Yasuda K., "On the Freedom in Covariance Control," Int. J. Control, to appear.

[13] Robust Control Toolbox for use with MATLAB, The MathWorks, Inc., 1988.

VARIANCE AND BIAS CONFIDENCE CRITERIA FOR SYSTEM AND OBSERVER/KALMAN FILTER IDENTIFICATION[1]

Dong-Huei Tseng[2] and Richard W. Longman[3]
Columbia University, New York, NY 10027

Jer-Nan Juang[4]
NASA Langley Research Center, Hampton, VA 23665

Chung-Wen Chen[5]
North Carolina State University, Raleigh, NC 27695

Abstract

Methods of estimating the uncertainty in identification results for the identified system, and for the identified steady-state Kalman filter, are developed for the Observer/Kalman Filter Identification (OKID) method. The method, which has proved very effective in a series of applications, is seen to have many advantages introduced by the identification of observer or Kalman filter Markov parameters as one of the steps in the identification process. The variance and bias results produced here, help the user determine how long a data record must be analyzed for a given accuracy in the identified parameters, and can help the user determine true system order, and eliminate noise modes. Computer programs have been developed to compute these variances, and a program tool box for general use is in preparation.

INTRODUCTION

In [1] several system identification algorithms related to the Ho Kalman algorithm are compared both theoretically and by numerical experiments, and the results suggest that the version of the eigensystem realization algorithm based on data correlation (ERA/DC) is to be preferred among those compared. People using such algorithms sometimes make Monte Carlo runs perturbing the data by an amount consistent with the expected measurement noise level, to see what identified system modes are well identified, and what modes might be modelling noise. Improving on this approach, in [2-6] perturbation theory methods were used to create estimates of the variance and bias of identified parameters based on a measured or assumed level of zero-mean white Gaussian noise in the Markov parameters supplied to the algorithm. This information not only gives the algorithm user estimates of the quality of his identification results, but they help him adjust parameters of the algorithm to obtain better results. In a separate set of papers, preprocessing of the data was introduced that first determines the Markov parameters of an observer or a Kalman filter, and this was shown to have important advantages [7-12]. The approach was termed OKID, referring to observer/Kalman filter identification. In the present paper, we develop variance and bias confidence criteria for identification results using OKID together with ERA/DC to identify the system and a Kalman filter directly from a long set of data. The OKID preprocessing is seen to have important advantages in the handling of noise. It also supplies, directly from the same set of data, the noise level information needed for the determination of the variances of the parameters, and it considers both plant and measurement noise in the modelling.

ERA/DC EQUATIONS AND SENSITIVITY ANALYSIS

The ERA/DC identification algorithm develops a continuous time model of the form

---

[1] Research supported by NASA Grant NAG 1-649.

[2] Graduate Research Assistant.

[3] Professor of Mechanical Engineering.

[4] Principal Scientist, Spacecraft Dynamics Branch.

[5] Research Associate, Department of Mechanical and Aerospace Engineering.

$$\begin{aligned}\dot{x}(t) &= A_c x(t) + B_c u(t)\\ y(kT) &= Cx(kT) + Du(kT)\end{aligned} \tag{1}$$

from input-output data. The equations involved are summarized here, and are used as part of the OKID algorithm discussed below. Experimental data are used corresponding to feeding the system by an input going through a zero order hold with sample time T. Then the continuous time system can be replaced by an equivalent difference equation model giving the same state and output histories at the sample times

$$\begin{aligned}x(k+1) &= Ax(k) + Bu(k)\\ y(k) &= Cx(k) + Du(k)\end{aligned} \tag{2}$$

(for simplicity we drop explicit mention of T in the notation). The input to the algorithm is the unit pulse response histories at times kT (the Markov parameters), which can be written in terms of the discrete time system matrices as $Y_{-1} = D$ for k=0, and $Y_{k-1} = CA^{k-1}B$ for k=1,2,3,.... This information could be directly measured experimentally, but is more commonly obtained by using fast Fourier transforms of rich input sequences, and of the corresponding output sequences, to obtain transfer functions between each input-output pair, and then taking their inverse transforms. The OKID algorithm substitutes a different procedure which has various advantages as discussed below. Then ERA/DC (in its simplest form) performs the following operations: Form Hankel matrices $H(0)$ and $H(1)$ according to

$$H(j) = \begin{bmatrix} Y_j & Y_{j+1} & \cdots & Y_{r+j}\\ Y_{j+1} & Y_{j+2} & \cdots & Y_{r+j}\\ \vdots & \vdots & & \vdots\\ Y_{s+j} & Y_{s+j+1} & & Y_{s+r+j}\end{bmatrix} \quad ; \qquad R_{DC}(0) = H(0)H^T(0) \tag{3}$$

and then perform the following operations including a singular value decomposition (where S contains only the nonzero eigenvalues of $R_{DC}(0)$) to obtain the discrete time realization

$$\begin{aligned}R_{DC}(0) &= U_1 S U_1^T\\ A &= S^{-1/2}U_1^T H(1)H^T(0)U_1 S^{-1/2}\\ B &= S^{-1/2}U_1^T H(0)E_m\\ C &= E_q^T U_1 S^{1/2}\end{aligned} \tag{4}$$

A continuous time realization is then obtained using an eigendecomposition (assuming that the system A matrix is diagonalizable, and that all frequencies of interest are within the Nyquist frequency so that we know which value of the complex logarithm to take)

$$\begin{aligned}&P^{-1}AP = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)\\ &\omega_j = \mathrm{Im}(\ln \lambda_j)/T \quad ; \quad \sigma_j = \mathrm{Re}(\ln \lambda_j)/T\\ &\gamma_j = \sigma_j + i\omega_j\\ &\Gamma = diag(\gamma_1, \gamma_2, \ldots, \gamma_n)\\ &A_c = P\Gamma P^{-1} \quad ; \quad B_c = A_c(A-I)^{-1}B\end{aligned} \tag{5}$$

The variance and bias estimates developed in [2-5], assume that there is additive zero-mean, white Gaussian noise n_k in the Markov parameter Y_k, for each k, and that one knows the covariance of this noise, call it π. For simplicity of notation assume here that these are all scalar quantities. Consider the Taylor series expansion of any identified parameter c as a function of this noise sequence n

$$c(n) = c(0) + \sum_k (\partial c / \partial n_k)\Big|_{n=0} n_k + (1/2)\sum_{j,k} (\partial^2 c / \partial n_j \partial n_k)\Big|_{n=0} n_j n_k + \cdots \tag{6}$$

Using this through linear terms, the variance of the identified parameter can be approximated by

$$\text{var}(c(n)) = E(c(n) - c(0))^2 = \sum_k [(\partial c / \partial n_k)\big|_{n=0}]^2 \pi \tag{7}$$

In practice, one does not have the information needed to evaluate the derivatives at n=0, and one approximates them by substituting these derivatives evaluated at the data used as input to the algorithm. A first approximation of the bias produced by sending zero mean Gaussian noise through a set of nonlinear equations is given by the difference between the expected value of the parameter through second order terms and the true value $c(0)$

$$\begin{aligned} E(c(n)) &= c(0) + (1/2)\sum_k (\partial^2 c / \partial n_k^2)\Big|_{n=0} \pi \\ Bias_q(c(n)) &= (1/2)\sum_k (\partial^2 c / \partial n_k^2)\Big|_{n=0} \pi \end{aligned} \tag{8}$$

which we term the quadratic bias. References [2-5] give other estimates as well, including truncation bias, and the uncertainty in the truncation bias, etc. All of the results assume a known noise level in the input Markov parameters. However, the results are very useful even without such information, since they show the sensitivity of the identification results to small changes in the data, and this is useful information. For example, it helps determine an appropriate model order by showing which modes of the identified system are insensitive to small changes in the data, and which modes are strongly influenced by such changes [6].

The main objective of this paper is to supply information about the variance and bias of system identification results obtained using the OKID procedure. A very important characteristic of the approach developed is that we will be able to determine directly from the given long sequence of input-output data, the noise level in the system. Thus, there is no need to know a noise level from some independent experiments or knowledge. Another advantage of the approach, is that it handles not only measurement noise, but also plant noise.

The method developed here makes use of the results obtained in [2-5] giving the first and second derivatives of various identified parameters (such as σ_j, ω_j, B_{ij}, and the singular values of the R_{DC} matrix) with respect to changes n_k in the Markov parameter inputs to the ERA/DC algorithm. The computation of these derivatives is non-trivial, and requires use of perturbation theory for eigenvalues, eigenvectors, singular values, and singular vectors. Due to space limitations, these equations will not be cited here.

WORLD MODELS

For the purposes of this paper where we wish to develop estimates of the uncertainties in identified system parameters using the OKID approach, we suppose that we can experimentally obtain the system output data sequence of $y(kT)$ that corresponds to a long string of rich inputs $u(kT)$ applied through a zero order hold, e.g. a random Gaussian white input sequence. We further assume that the real world that produced the data can be represented in the form of a finite dimensional linear time-invariant system

$$\begin{aligned} \dot{x}(t) &= A_c x(t) + B_c u(t) + w_c(t) \\ y(kT) &= Cx(kT) + Du(kT) + v(kT) \end{aligned} \tag{9}$$

where the w_c and v are additive stationary, zero-mean, white and Gaussian, plant and measurement noises, with covariances Q_C and R, respectively.

A second world model in the form of a discrete time representation can be obtained without any approximation, since the input goes through a zero-order hold:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + Du(k) + v(k) \end{aligned} \tag{10}$$

Here the new plant noise has covariance Q given by $Q = P\overline{Q}P^T$ and

$$\overline{q}_{ij} = (\gamma_i + \gamma_j)^{-1}[\exp((\gamma_i + \gamma_j)T) - 1]\overline{q}_{cij} \tag{11}$$

where $\overline{q}_{ij}$ and $\overline{q}_{cij}$ are the elements of $\overline{Q}$ and $\overline{Q}_c = P^{-1}Q_c P^{-T}$, respectively. The A_C, B_C matrices are related to the A, B matrices as above, and the relationship is invertible, provided one knows that all system modes have frequencies below the Nyquist frequency for the sample rate used so that one knows what imaginary part to ascribe to the complex logarithm in (5).

A third world model is the steady-state, one-step ahead Kalman filter equation for system (9) (we will also call this an observer) which can be written in the form

$$\begin{aligned} \hat{x}(k+1) &= \tilde{A}\hat{x}(k) + \tilde{B}v(k) \\ y(k) &= C\hat{x}(k) + Du(k) + \varepsilon(k) \end{aligned} \tag{12}$$

where

$$\tilde{A} = A - KC \;\; ; \;\; \tilde{B} = \left[B - KD, K\right] \;\; ; \;\; v(k) = \begin{bmatrix} u(k) \\ y(k) \end{bmatrix} \tag{13}$$

In these equations, K is the one-step ahead, steady-state Kalman gain ($A^{-1}K$ is the steady-state Kalman gain of the true-filter), and $\varepsilon(k)$ is the innovations process

$$\begin{aligned} \varepsilon(k) &= y(k) - \hat{y}(k) \\ \hat{y}(k) &= C\hat{x}(k) + Du(k) \end{aligned} \tag{14}$$

Note that in this model, there is "measurement noise", but no plant noise. Furthermore, the measurement noise is the innovations process; under the assumption made above that the system producing the data can be represented by equation (9), we conclude that the measurement noise in this world model is white, zero-mean, and Gaussian.

FINDING THE OBSERVER MARKOV PARAMETERS

Given a long sequence of ℓ input-output pairs for k = 0, 1, 2, ..., ℓ-1 in system (12), the set of equations (12) relating these for each time step can be written in matrix form as

$$\overline{y} = \tilde{Y}\overline{V} + \overline{\varepsilon} + C\tilde{A}^p\,\hat{\mathrm{x}} \tag{15}$$

where

$$\bar{y} = \left[y(p+1) \quad y(p+2) \quad \cdots \quad y(\ell-1) \right]$$

$$\tilde{Y} = \left[D \quad C\tilde{B} \quad C\tilde{A}\tilde{B} \quad \cdots \quad C\tilde{A}^{p-1}\tilde{B} \right]$$

$$\bar{\varepsilon} = \left[\varepsilon(p+1) \quad \varepsilon(p+2) \quad \cdots \quad \varepsilon(\ell-1) \right]$$

$$\hat{\mathrm{x}} = \left[\hat{x}(0) \quad \hat{x}(1) \quad \hat{x}(2) \quad \cdots \quad \hat{x}(\ell-p-2) \right] \tag{16}$$

$$\bar{V} = \begin{bmatrix} u(p+1) & u(p+2) & \cdots & u(\ell-1) \\ v(p) & v(p+1) & \cdots & v(\ell-2) \\ v(p-1) & v(p) & \cdots & v(\ell-3) \\ \vdots & \vdots & \ddots & \vdots \\ v(1) & v(2) & \cdots & v(\ell-p-1) \end{bmatrix}$$

When p is picked sufficiently large, the term containing $\tilde{A}^p$ becomes negligible and can be dropped. One of the important advantages of solving for the observer Markov parameters for equation (12), instead of the Markov parameters of the system (10), is that the eigenvalues of matrix $\tilde{A}$ are those of the Kalman filter, which are usually much more stable than those of system (10) itself, allowing a smaller value of p. This can be important in lightly damped systems such as lightly damped spacecraft structures, and is essential in unstable systems. Other important advantages relate to the handling of noise by the method, the ability to evaluate the noise level as is done in this paper, and the ability to find the Kalman gain, as discussed below. After dropping this term, a least squares solution of equation (15) is used

$$\tilde{Y}_e = \bar{y}\bar{V}^T[\bar{V}\bar{V}^T]^{-1} = \bar{y}\bar{V}^+ \tag{17}$$

giving an approximate solution for the observer/Kalman filter Markov parameters $\tilde{Y}$ in the presence of noise.

TWO IDENTIFICATION APPROACHES

Given the observer Markov parameters obtained above, there are two approaches we could use to identify the system matrices A, B, C, D, from which we could obtain the continuous time model A_C, B_C, C, D as described above. We might also be interested in knowing the Kalman gain K, and to the extent possible, the plant and measurement noise levels in the various world models described above.

Approach 1

Given the observer Markov parameters $D, C\tilde{B}, C\tilde{A}\tilde{B}, C\tilde{A}^2\tilde{B}, \ldots$ one can use ERA/DC to determine matrices $\tilde{A}, \tilde{B}, C, D$. Knowledge of $\tilde{B}$ means that we know K by the second of equations (13), and then knowledge of K and D means that we know B from the same equation in (13), and then knowledge of K and C means that we know A from the first of (13). This approach is necessary if one wants to know the Kalman gain K. The second approach, given below, will be seen to have advantages when we are only interested in a system model.

Approach 2

In this approach, instead of using the observer Markov parameters to find the observer matrices from which we find the system matrices, we use the observer Markov parameters to find the system Markov parameters, and from these we find the system matrices by ERA/DC. It was shown in [11] and [7-10], that the system Markov parameters Y_k defined below equation (2) can be obtained from the observer Markov parameters by a recursive computation

$$Y_k = \tilde{Y}_k^{(1)} + \sum_{i=0}^{k-1} \tilde{Y}_i^{(2)} Y_{k-i-1} + \tilde{Y}_k^{(2)} D \tag{18}$$

where the partitions of the observer Markov parameters are defined as follows

$$\begin{aligned} \tilde{Y}_k &= C\tilde{A}^k \tilde{B} \\ &= \left[C(A-KC)^k (B-KD),\ C(A-KC)^k K \right] \\ &\equiv \left[\tilde{Y}_k^{(1)},\ \tilde{Y}_k^{(2)} \right];\ k = 0,1,2,\ldots \end{aligned} \tag{19}$$

UNBIASED ESTIMATION OF THE OBSERVER MARKOV PARAMETERS

In equation (15) there is the explicitly written noise term $\bar{\varepsilon}$, but also both $\bar{y}$ and $\bar{V}$ contain noisy data, and in fact the same noises appear in each. This is a common situation in least squares problems, and it normally produces bias in the least squares estimates. However, we are estimating the Kalman filter Markov parameters, and this allows us to establish that the least squares solution (17) produces zero-mean, and white equation errors that are orthogonal to the output data. Assume that p has been picked so that the last term in (15) is negligible. Then define the equation error sequence ε_e for equation (15) that corresponds to the least squares solution $\tilde{Y}_e$:

$$\begin{aligned} \bar{\varepsilon}_e &= \bar{y} - \tilde{Y}_e \bar{V} \\ \bar{\varepsilon}_e &= \left[\varepsilon_e(p+1) \quad \varepsilon_e(p+2) \quad \cdots \quad \varepsilon_e(\ell-1) \right] \end{aligned} \tag{20}$$

From the first of these we can write

$$\begin{aligned} &\bar{y}\bar{V}^T - \tilde{Y}_e \bar{V}\bar{V}^T = \bar{\varepsilon}_e \bar{V}^T \\ &= \left[\sum_{k=p+1}^{\ell-1} \varepsilon_e(k) u^T(k) \quad \sum_{k=p+1}^{\ell-1} \varepsilon_e(k) v^T(k-1) \quad \cdots \quad \sum_{k=p+1}^{\ell-1} \varepsilon_e(k) v^T(k-p) \right] \end{aligned} \tag{21}$$

By definition of the least squares solution, the left hand side of the equation is zero. Hence, we conclude that

$$\begin{aligned} \frac{1}{\ell-p-1} \sum_{k=p+1}^{\ell-1} \varepsilon_e(k) y^T(k-i) = 0 \quad &; \quad i = 1,2,\ldots,p \\ \frac{1}{\ell-p-1} \sum_{k=p+1}^{\ell-1} \varepsilon_e(k) u^T(k-j) = 0 \quad &; \quad j = 0,1,\ldots,p \end{aligned} \tag{22}$$

The first of these equations is a finite version of the orthogonality principle that a Kalman filter must satisfy so that it produces optimal estimates. The fact that such a finite version of the orthogonality principle is satisfied for all data lengths is an important feature, that helps to produce good results. The second equation, as $\ell \to \infty$ says that the residuals are zero mean, because we are at liberty to choose any sequence we wish for the inputs u. Below we show that the residuals approach the true noise as $\ell \to \infty$, and then these results applied successively to the expression (12) shows that the residuals become white in the limit as $\ell \to \infty$. In particular,

$$\begin{aligned} &E[\varepsilon_e(k)y^T(k-i)]=0 \quad ; \quad i=1,2,...,p \quad ; \quad k>p \\ &E[\varepsilon_e(k)]=0 \\ &E[\varepsilon_e(k)\varepsilon^T(k-j)]=0 \quad ; \quad j>0 \end{aligned} \tag{23}$$

The fact that we are applying the least squares procedure to the Kalman filter system rather than a model of the actual system to be identified, allows us to package the plant noise as if it were measurement noise. This fact allows us to get unbiased estimates of the observer Markov parameters. Thus, the OKID process of computing observer Markov parameters from the data as an intermediate step in producing the identification results, is a process that whitens the residuals when the data comes from a system of the form (9) consisting of a finite order system with white additive plant and measurement noises. This is the most important feature of OKID. Although, so far we have only discussed getting the observer Markov parameters, as the data length gets large, the observer Markov parameters will converge to their true values as seen below, so that the ERA/DC identified system is an unbiased identification as well.

EVALUATING THE NOISE LEVEL -- THE INNOVATIONS COVARIANCE

It is of interest to relate the residual error sequence for the least squares fit $\overline{\varepsilon}_e$ to the actual innovations sequence $\overline{\varepsilon}$. Substituting equation (15) into (17) produces an expression for the error in the identified observer Markov parameters denoted by n

$$\begin{aligned} \tilde{Y}_e &= \overline{y}\overline{V}^+ = (\tilde{Y}\overline{V} + \overline{\varepsilon})\overline{V}^+ = \tilde{Y} + \overline{\varepsilon}\overline{V}^+ \\ \tilde{Y}_e - \tilde{Y} &= \overline{\varepsilon}\overline{V}^+ = n \end{aligned} \tag{24}$$

Equating the $\overline{y}$ in (20) and (15), and substituting the first of (24) produces the desired relationship

$$\overline{\varepsilon}_e = \overline{\varepsilon}(I - \overline{V}^+\overline{V}) \tag{25}$$

Write the singular value decomposition $\overline{V} = U_s[S \quad 0]V_s^T$ were the first and last matrices are unitary, and the last one in partitioned associated with the dimension of S as $V_s = [V_{s1} \quad V_{s2}]$. The matix V_s can be thought of as an orthonormal basis of the the space containing $\overline{\varepsilon}$ and $\overline{\varepsilon}_e$. Note that V_{s1} forms a basis of a small dimensional subspace with dimension associated with the choice of p. The V_{s2} spans a large dimensional subspace with dimension related to the length of the data record, and this dimension goes to infinity as the length of the data record goes to infinity. Then substitution into (25) gives $V_{s2}^T\overline{\varepsilon}_e = V_{s2}^T\overline{\varepsilon}$ and $V_{s1}^T\overline{\varepsilon}_e = 0$, saying that $\overline{\varepsilon}$ and $\overline{\varepsilon}_e$ have identical components in the large dimensional space of V_{s2} and that they only differ in that the components of $\overline{\varepsilon}$ in the small dimensional subspace of V_{s1} are incorrectly represented as zero in $\overline{\varepsilon}_e$. As the data length becomes large, this error becomes negligible.

Since the plant and measurement noises were assumed to be stationary random processes, we can use the ergotic property to equate the covariance of the innovations term to the infinite time average of the square of the innovation history. We approximate this by a finite time average and substitute $\overline{\varepsilon}_e$ for $\overline{\varepsilon}$ to obtain the approximate innovations covariance

$$\Pi \equiv E(\varepsilon(k)\varepsilon^T(k)) \approx (1/d)\overline{\varepsilon}_e\overline{\varepsilon}_e^T \quad ; \quad d = \ell - p - 1 \tag{26}$$

INFORMATION ABOUT THE PLANT AND MEASUREMENT NOISE AND THE STATE UNCERTAINTY

In Approach 1 above, the identification produces the matrices A_C, B_C, C, D, which gives a continuous time model of the system. We also obtain the additional information of the Kalman gain K, so that we know the Kalman filter to use in the system, for example in a control system design. And from (26) we know the innovations covariance Π, which is an important measure of the noise level in the system. However, it might be desirable to know four more quantities related to the continuous time and the discrete time system models and the Kalman filter: the plant noise convariances Q_c and Q, the measurement noise covariance R, and the steady state covariance of the Kalman filter state estimate P_x. The equations governing these quantities are

$$\begin{aligned} \Pi &= CP_xC^T + R \\ P_x &= \tilde{A}P_x\tilde{A}^T + Q + KRK^T \\ K &= P_xC^T\Pi^{-1} \\ P_x &\equiv E[(x(k)-\hat{x}(k))(x(k)-\hat{x}(k))^T] \end{aligned} \tag{27}$$

The first of these expressions evaluates the innovations covariance in terms of the plant and measurement covariances, and is obtained by solving the second of (12) for $\varepsilon(k)$, substituting the $y(k)$ from (10), and using this to compute the covariance. The second equation uses the invariance of the covariance P_x to replace the k in the definition of P_x in (27) by $k+1$, then substitutes the state equation (10) and the Kalman filter (12) to arrive at the result. The third equation gives the steady state Kalman gain, and is obtained by satisfying the orthogonality condition $E[(x(k)-\hat{x}(k))y^T(k)] = 0$ for optimality of the filter.

The first and third equation in (27) can be thought of as simultaneous equations for R and P_x, and the second equation is then an equation for Q. Note that if Q is determined, then one can obtain Q_c uniquely, as described by equation (11). The measurement noise covariance R is uniquely determined from our knowledge of Π, K and C

$$R = (I - CK)\Pi \tag{28}$$

To understand the information available concerning P_x and Q, it is instructive to make a change of variables from x to z, based on the singular value decomposition of C,

$$x = V_c diag(S_c^{-1}U_c^T, I)z \quad ; \quad C = U_c[S_c \quad 0]V_c^T \tag{29}$$

so that the new output influence matrix is $C' = [I \quad 0]$, which has the property that the first partition of the new state variables is directly measured, and second partition corresponds to states that an observer would have to reconstruct. In this new coordinate frame, denote the partitions of the new P_x matrix as P_{11}, P_{21}, and P_{22}, with P_{12} the transpose of P_{21}. Then the third equation in (27) gives

$$\begin{bmatrix} P_{11} \\ P_{21} \end{bmatrix} = K\Pi \tag{30}$$

and this is the only information available about P. In particular, there is no information about P_{22}, i.e., there is no information about the covariance of the state variables that were not measured, as defined by C'. The second of equations (27), which is the only information we have about Q (in the z coordinates), can now be written as

$$\begin{bmatrix} Q_{11} + \tilde{A}_{12}P_{22}\tilde{A}_{12}^T & Q_{21}^T + \tilde{A}_{12}P_{22}\tilde{A}_{22}^T \\ Q_{21} + \tilde{A}_{22}P_{22}\tilde{A}_{12}^T & Q_{22} + \tilde{A}_{22}P_{22}\tilde{A}_{22}^T - P_{22} \end{bmatrix} = -\tilde{A}\begin{bmatrix} P_{11} & P_{21}^T \\ P_{21} & 0 \end{bmatrix}\tilde{A}^T + \begin{bmatrix} P_{11} & P_{21}^T \\ P_{21} & 0 \end{bmatrix} - KRK^T \tag{31}$$

The right hand side of the equation is known, allowing us to know the combinations of Q and P_{22} that appear on the left, but not Q itself. Since the P_{22} is totally unknown, there is no way to solve for Q, and furthermore all partitions of Q are in general affected by P_{22} on the left of (31), so that we do not even know any partitions of Q as we did for P.

These results indicate that a long sequence of rich input-output data contains enough information to determine the measurement noise covariance R, but does not contain enough information to determine the steady state covariance of Kalman filter estimates of unmeasured state variables, and does not contain enough information to know the plant noise covariance. There are an infinite number of corresponding solutions for each, that allow exchange between the uncertainties represented by Q and those represented by P_{22}. These uncertainties, however, are of no consequence, if the aim of the identification is to be able to predict the response of measured variables, or to design a controller for controlling these variables.

VARIANCE ESTIMATION FOR IDENTIFIED PARAMETERS, MARKOV PARAMETERS, KALMAN GAIN, AND SINGULAR VALUES

With the information developed above, it is possible to develop estimates of the uncertainties in the identification results. These estimates are described below. A tool box computer program package is under development that allows one to compute the variances discussed below. This not only tells one how much confidence to place in identification results, but it also allows one to tune parameters in the ERA/DC algorithm to get better results, and it tells one how much data must be used to obtain a certain level of confidence in the results.

The General Variance Formula

Assuming that we have a reasonable estimate of the innovations covariance Π, we now extend the perturbation analysis summarized in equations (6-8), to estimate the variance of any desired identified parameter, which we denote by c. The value of the identified parameter is a function of the noise n in the observer Markov parameters, which in turn is a function of the innovations history $\bar{\varepsilon}$ according to $n = \bar{\varepsilon}\bar{V}^{+}$, so that $c = c(\bar{\varepsilon})$, or more specifically $c = c(n(\bar{\varepsilon}))$. Denote by $\bar{V}_o^{+}$ the value of $\bar{V}^{+}$ evaluated at $\bar{\varepsilon} = 0$. Then the change in parameter c due to changes $\bar{\varepsilon}$ in the data, when linearized about the true solution, can be written as

$$
\begin{aligned}
c(\bar{\varepsilon}) \cong c(0) + \sum_{k=p+1}^{\ell-1} \left[\partial c / \partial \varepsilon(k)\big|_{\bar{\varepsilon}=0}\right]^T \varepsilon(k) \\
\partial c / \partial \bar{\varepsilon}\big|_{\bar{\varepsilon}=0} = \left[\partial c / \partial n\big|_{\bar{\varepsilon}=0}\right]\left[\bar{V}_o^{+}\right]^T
\end{aligned}
\tag{32}
$$

where the chain rule has been used to evaluate the derivative, and the *i,j* component of a scalar differentiated with respect to a matrix is defined as the derivative of the scalar with respect to the *i,j* component of that matrix. As before, we do not know the values of the derivatives evaluated at the true solution, so that we substitute approximate values obtained as the derivatives evaluated at the solution that we obtain in the identification. The variance of this linear relationship in terms of the innovations covariance Π is

$$
\begin{aligned}
E[(c(\bar{\varepsilon}) - c(0))(c(\bar{\varepsilon}) - c(0))^T] = \sum_{k=p+1}^{\ell-1} \left[\partial c / \partial \varepsilon(k)\big|_{\bar{\varepsilon}=0}\right]^T \Pi \left[\partial c / \partial \varepsilon(k)\big|_{\bar{\varepsilon}=0}\right] \\
= \sum_{i=1}^{\ell-p-1} v_i \left[\partial c / \partial n\big|_{n=0}\right]^T \Pi \left[\partial c / \partial n\big|_{n=0}\right] v_i^T = trace\left\{ \sum_{i=1}^{\ell-p-1} v_i^T v_i \left[\partial c / \partial n\big|_{n=0}\right]^T \Pi \left[\partial c / \partial n\big|_{n=0}\right] \right\}
\end{aligned}
\tag{33}
$$

where v_i is the ith row of $\bar{V}_o^{+}$, and whiteness has been used on the first line. This can be rewritten as the following result which we will use for many purposes in the following sections

$$
E[(c(\bar{\varepsilon}) - c(0))(c(\bar{\varepsilon}) - c(0))^T] = trace\left\{ (\bar{V}_o^{+})^T (\bar{V}_o^{+}) \left[\partial c / \partial n\big|_{n=0}\right]^T \Pi \left[\partial c / \partial n\big|_{n=0}\right] \right\}
\tag{34}
$$

The $(\bar{V}\bar{V}^T)^{-1}$ Matrix as a Function of Data Length

The pseudoinverses in the trace in (34) are known from the solution for the observer Markov parameters in equation (17). This product in the trace has important properties. Note that

$$(\bar{V}_o^+)^T(\bar{V}_o^+) = (\bar{V}\bar{V}^T)^{-1} \tag{35}$$

The *i,j* partition of the product $\bar{V}\bar{V}^T$ can be written as

$$d\{(1/d)\sum_{k=p+1}^{\ell-1} v(k-i)v^T(k-j)\}$$

where *i,j* go from 0 to p, the $v(k-\alpha)$ is replace by $u(k)$ when α is zero, and $d=\ell-p-1$. Note that if for example we choose to apply control inputs that are picked to form a stationary, white zero-mean Gaussian sequence, then the expression in the brackets will converge to a steady-state value as the data length ℓ is increased, corresponding to the covariance of v. And in processing the data, one can watch the convergence of this matrix. Once the value of $(1/d)\bar{V}\bar{V}^T$ has stabilized, then the variance of the parameter c given by equation (34) becomes a monotonically decreasing function of the data length ℓ as d increases. This allows one to make decisions about how many data points to use to obtain a desired level of uncertainty.

The Variance of the Observer Markov Parameters

The easiest application of the formula (34) is to determine the variance of the identified observer Markov parameters. One lets c be any scalar element of such a parameter, and then the derivatives needed on the right hand side of the equation are all very simple, either zero or one. The observation above about choice of the data length can be used to get a desired accuracy of these parameters, and once these parameters are accurately known, the rest of the identification will give good results.

The Variance of the System Markov Parameters

When using Approach 2, the actual input to the ERA/DC identification algorithm is the system Markov parameters, not the observer Markov parameters, and the variance of these parameters would be more relevant. Setting c to be any desired element of these parameters, the derivatives needed on the right of (34) become more complicated, requiring use of the recursive relationship

$$\frac{\partial Y_k}{\partial n_{ij}} = \frac{\partial \tilde{Y}_k^{(1)}}{\partial n_{ij}} + \sum_{i=0}^{k-1}\left(\frac{\partial \tilde{Y}_i^{(2)}}{\partial n_{ij}} Y_{k-i-1} + \tilde{Y}_i^{(2)}\frac{\partial Y_{k-i-1}}{\partial n_{ij}}\right) + \frac{\partial \tilde{Y}_k^{(2)}}{\partial n_{ij}} D + \tilde{Y}_k^{(2)}\frac{\partial D}{\partial n_{ij}} \tag{36}$$

together with equation (18).

The Uncertainty in the Identified Frequency and Damping of System Modes

The eigenvalues of the identified system are often important quantities physically, and they are invariant with respect to changes in the realization used. The variances of these eigenvalues are good indicators of identification accuracy. The original motivation for the development of the ERA algorithm, and for the development of the perturbation methods in [2-5] for assessing accuracy, is identification of vibrations in structures such as flexible spacecraft. In these applications the imaginary parts of the continuous system eigenvalues are the vibration frequencies of the structure and the real parts are the damping of the vibration mode. The derivatives of these σ_j, ω_j, with respect to changes in the system Markov parameters have been developed in [2-5] for the ERA/DC algorithm. Then by use of the chain rule, one combines these derivative with those developed using (36) above, in order to obtain the variances of these quantities using the variance formula (34). Here we are using Approach 2 which is the preferred approach as discussed below.

The Uncertainty in the Kalman Gain

In order to obtain the Kalman gain, one must use Approach 1, and then the gain is given by the second partition of the identified input influence matrix $\tilde{B}$ in (13). The derivative of the input influence matrix with respect to changes in the input Markov parameters is also available from [2-5], and this gives directly the derivatives needed to compute the variances for each element of the Kalman gain.

The Variance of the Singular Values

The perturbation theory methods in [2-5] also compute other quantities such as the variance of the singular values of the matrix $R_{DC}(0)$ in equation (4). These singular values are considered useful indicators of the true system order. When the noise level is sufficiently small, then there is a substantial drop between the singular values associated with the system, and singular values associated with noise. However, it is always possible to increase the noise level sufficiently that the drop disappears. It is shown in [2-5] that the variance of the singular values is a more sensitive indicator of system order than the singular values themselves. Other system order indicators are also discussed in these references, and can apply here. Use of the variances of the mode frequencies and damping factors to determine order is discussed in [6].

BIAS ESTIMATION

When ERA/DC is used directly on noisy system Markov parameters, without using the OKID procedure, then the noise going through the nonlinear identification equations produces biased results. It is typical for these biases to increase as the amount of data used is increased. A major advantage of the OKID procedure discussed here, is that as the data length increases the results become unbiased, as we have shown. Nevertheless, for finite data lengths, one may be interested in estimating the bias due to nonlinearity of the ERA/DC equations, and of the conversion equations between observer Markov parameters and system Markov parameters. The first approximation to this bias comes from the expected value of the second order terms in the Taylor series expansion, adding the next term in equation (32). These terms do not average to zero because they contain the square of zero-mean noise, which is not zero mean. This approximation to the bias, which we term the quadratic bias, can be estimated using

$$\begin{aligned} Bias_q(c(\bar{\varepsilon})) &= \sum_{i,j,k,l} E(\bar{\varepsilon}_{ij}\bar{\varepsilon}_{kl})(1/2)(\partial^2 c/\partial\bar{\varepsilon}_{ij}\partial\bar{\varepsilon}_{kl})\Big|_{\bar{\varepsilon}=0} \\ &= (1/2)\sum_{i,j} trace\left\{(\bar{V}_o^+)^T(\bar{V}_o^+)\Pi_{ij}(\partial^2 c/\partial n_i \partial n_k)\Big|_{\bar{\varepsilon}=0}\right\} \end{aligned} \tag{37}$$

Here the α,β element of the second derivative matrix with respect to the *i*th and *j*th columns of *n* is defined to be the derivative with respect to the αth element of the *i*th column and the βth element of the *j*th column. Whiteness of the noise has again been used. Note that the matrix $(\bar{V}\bar{V}^T)^{-1}$ plays a fundamental role in the size of the bias as well as the variance, and that after the number of data points is sufficiently large, the quadratic bias becomes a monotonically decreasing function of the data length in *d* because of its dependence on $1/d$.

COMPARISON OF THE APPROACHES

Approach 1 is the only option available when one wants to know the Kalman gain, but if one is only interested in identifying the given system, both Approaches 1 and 2 could be used. Looking at the variance formula (34), one might think that the two approaches simply offer two different ways to calculate the parameter *c* (which could, for example, be an eigenvalue of the identified system) from the given Markov parameters, and hence, the derivatives of the parameter *c* with respect to changes *n* in the Markov parameters would be the same. However, this is not true. The derivatives with respect to elements of n are different. If one were to change all of the Markov parameters in a manner that could be represented by products of matrices of the form $C\tilde{A}^k\tilde{B}$, then both methods much come up with the same answer. But when the changes in the Markov parameters are not ones that can be modelled exactly, the methods are free to give different answers. Of course, in the limit as $\ell \rightarrow \infty$, both methods are unbiased and must converge to the same answer. What is observed in practice, is that Approach 1 takes longer to converge to good identified values than Approach 2. Thus, Approach 2 is recommended unless one needs to determine the Kalman gain.

REFERENCES

1. J.-S. Lew, J.-N. Juang, and R. W. Longman, "Comparison of Several System Identification Methods for Flexible Structures," *Proceedings of the 32nd AIAA Structures, Structural Dynamics and Materials Conference*, Baltimore, MD, April 1991.
2. R. W. Longman, J.-S. Lew, D.-H. Tseng, and J.-N. Juang, "Variance and Bias Computation for Improved Modal Identification Using ERA/DC," *Proceedings of the American Control Conference*, Boston, Massachusetts, May 1991.
3. R. W. Longman and J.-N. Juang, "A Variance Based Confidence Criterion for ERA Identified Modal Parameters," *Advances in the Astronautical Sciences*, American Astronautical Society, Vol. 65, Part I, 1988, pp. 581-601.
4. R. W. Longman, M. Bergmann and J.-N. Juang, "Variance and Bias Confidence Criteria for ERA Modal Parameter Identification," *Proceedings of the AIAA/AAS Astrodynamics Conference,* Minneapolis, Minnesota, August 1988, pp. 729-739.
5. M. Bergmann, R.W. Longman, and J.N. Juang, "Variance and Bias Computation for Enhanced System Identification," *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, Dec. 1989, pp. 155-161.
6. R. W. Longman, J. S. Lew, and J. N. Juang, "Comparison of Candidate Methods to Distinguish Noise Modes from System Modes in Structural Identification," *Proceeding of the AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference (SDM)*, Dallas, Texas, April 1992.
7. Phan, M., Juang, J.-N., Longman, R. W., "Identification of Linear Systems from a Single Set of Data by Identification of Observers with Assigned Real Eigenvalues," *Proceedings of the AIAA 32nd Structures, Structural Dynamics, and Materials Conference*, Baltimore, Maryland, April 1991. Accepted for publication in *The Journal of the Astronautical Sciences.*
8. Phan, M., Horta, L. G., Juang, J.-N., Longman, R. W., "Linear System Identification Via an Asymptotically Stable Observer," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, August 1991.
9. Phan, M., Horta, L. G., Juang, J.-N., Longman, R. W., "Identification of Linear Systems Via an Asymptotically Stable Observer," NASA Technical Report, to appear.
10. C.-W. Chen, J.-K. Huang, M. Phan, J.-N. and Juang, "Integrated System Identification and Modal State Estimation for Control of Large Flexible Space Structures," *Proceedings of the AIAA Guidance, Navigation, and Control Conference,* Portland, Oregon, August 1990, pp. 1396-1404. Also accepted for publication in the *Journal of Guidance, Control and Dynamics.*
11. J.-N. Juang, L. G. Horta, M. Phan, and R. W. Longman, "Identification of Observer/Kalman Filter Markov Parameters: Theory and Experiment," *Proceedings of the 1991 AIAA Guidance, Navigation and Control Conference*, New Orleans, Louisiana, August 1991, to appear in *Journal of Guidance, Control and Dynamics.*
12. M. Phan, L. G. Horta, J.-N. Juang, and R. W. Longman, "Improvement of Observer/Kalman Filter Identification by Residual Whitening," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Hilton Head, South Carolina, August 1992.

THE DETERMINATION OF FEEDBACK GAINS FOR UNCERTAIN LINEAR SYSTEMS USING HIGHER-ORDER SENSITIVITY MINIMIZATION

Brian C. Fabien, Assistant Professor, Department of Mechanical Engineering
Ohio University, Athens Ohio 45701

ABSTRACT

To reduce the influence of large parameter variation on the response of linear systems this paper considers the design of controllers that minimize the higher-order sensitivities. Here, the higher-order sensitivities are defined as the second-order, third-order, etc., partial derivatives of the state and control vectors with respect to the uncertain parameter. The feedback gains are obtained by minimizing a quadratic cost function that includes state and control sensitivities of arbitrary order.

PROBLEM AND SOLUTION

One approach to the design of linear feedback controllers for uncertain systems is the trajectory sensitivity minimization technique. In this approach the controller gains are selected to minimize the trajectory sensitivity, i.e., the first-order partial derivative of the trajectory (state variable) with respect to the uncertain parameter. As a result, small parameter perturbations have a reduced influence on the behavior of the system. The validity of this approach relies on the assumption that the parameter variation always remains small. If the parameter variations are large however, higher-order sensitivities must be considered.

Consider a LTI system that depends on a scalar parameter α such that

$$\dot{\bar{\mathbf{x}}}(t) = \bar{\mathrm{A}}(\alpha)\bar{\mathbf{x}}(t) + \bar{\mathrm{B}}(\alpha)\mathbf{u}(t) + \bar{\mathrm{D}}(\alpha)\mathbf{w}(t) \qquad (1)$$

$$\mathbf{y}(t) = \bar{\mathrm{C}}(\alpha)\bar{\mathbf{x}}(t) \qquad (2)$$

where the state vector $\bar{\mathbf{x}}(t) \in R^{n}$, the control vector $\mathbf{u}(t) \in R^{m}$, the output vector $\mathbf{y}(t) \in R^{p}$, and the disturbance input vector $\mathbf{w}(t) \in R^{d}$. The disturbance input is assumed to be a white, gaussian, zero mean, random process with intensity, $E(\mathbf{w}(t)\mathbf{w}(\tau)^{T}) = \mathrm{W}\delta(t-\tau)$, $\mathrm{W} \in R^{d \times d}$ and $\mathrm{W} \geq 0$. Here, $E(\bullet)$ denotes the expected value. The input disturbance is assumed to be independent of the parameter α. In order to simplify the notation the explicit dependence of $\bar{\mathbf{x}}$, $\mathbf{u}$, $\mathbf{w}$ and $\mathbf{y}$ on the time, t, is not shown. Similarly, the explicit dependence of $\bar{\mathrm{A}}$, $\bar{\mathrm{B}}$, $\bar{\mathrm{C}}$ and $\bar{\mathrm{D}}$ on the parameter α is not shown. The matrices $\bar{\mathrm{A}}$, $\bar{\mathrm{B}}$, $\bar{\mathrm{C}}$ and $\bar{\mathrm{D}}$ are of appropriate dimensions. It is assumed that the partial derivatives (with respect to the parameter α) of the matrices $\bar{\mathrm{A}}$, $\bar{\mathrm{B}}$, $\bar{\mathrm{C}}$ and $\bar{\mathrm{D}}$ exist and are continuous. The pair $(\bar{\mathrm{A}},\bar{\mathrm{B}})$ is controllable, and the pair $(\bar{\mathrm{A}},\bar{\mathrm{C}})$ is observable for all values of α.

The problem addressed in this paper is that of finding a constant gain output feedback control $\mathbf{u} = \mathrm{L}\mathbf{y}$ that minimizes the cost function

$$\bar{J} = \lim_{t\to\infty} E\left(\bar{\mathbf{x}}^{T}\bar{\mathrm{Q}}_{0}\bar{\mathbf{x}} + \mathbf{u}^{T}\mathrm{R}_{0}\mathbf{u} + \sum_{i-1}^{n_s}\left(\bar{\mathbf{x}}_i^{T}\bar{\mathrm{Q}}_i\bar{\mathbf{x}}_i + \mathbf{u}_i^{T}\mathrm{R}_i\mathbf{u}_i \right) \right) \qquad (3)$$

where $\bar{\mathrm{Q}}_i = \bar{\mathrm{Q}}_i^{T} \geq 0$, $\mathrm{R}_i = \mathrm{R}_i^{T} \geq 0$, $i = 0, 1, \ldots, n_s$. Here, $\bar{\mathbf{x}}_i = \partial^{i}\bar{\mathbf{x}}/\partial\alpha^{i} \in R^{n}$ is defined as the ith order trajectory sensitivity and $\bar{\mathbf{u}}_i = \partial^{i}\bar{\mathbf{u}}/\partial\alpha^{i} \in R^{m}$ is defined as the ith order control sensitivity. The controller gain matrix, L, is assumed to be independent of the parameter α.

The first and second terms in equation (3) are quadratic functions of the state and control vectors, respectively. The third and fourth terms are a sum of quadratic functions of the ith order trajectory sensitivity and the ith order control sensitivity, respectively. Hence, the cost function includes (trajectory and control) sensitivities of order 1 through n_s.

The standard Linear Quadratic (LQ) cost function can be recovered from equation (3) by selecting $\bar{Q}_0 \geq 0$, $R_0 > 0$, $\bar{Q}_i = 0$, and $R_i = 0$, $i = 1, 2, ..., n_s$. The standard LQ cost function can be used to obtain optimal performance of the nominal system at $\alpha = \alpha_0$.

Emphasis can be placed on reducing the i^{th} order sensitivity alone by appropriate selection of $\bar{Q}_i$, R_i, $i = 1, 2, ..., n_s$, with $\bar{Q}_0 = 0$ and $R_0 = 0$. It should be noted that to have a well-defined optimal control problem and to ensure finite controller gains at least one R_i ($i = 0, 1, ..., n_s$) must be positive definite. The cost function given by equation (3) with $n_s = 1$ was used by Yedavalli and Skelton,[1] and Okada and Skelton[2] in the design of controllers that yield closed-loop systems with reduced parameter sensitivity. In this paper sensitivities of order $n_s \geq 1$ are minimized.

It can be shown that the first-order necessary conditions for the controller gain matrix, L, to be optimal in the sense of (3) are

$$A^T P + PA + Q = 0; \quad A\Lambda + \Lambda A^T + DWD^T = 0;$$

$$2\sum_{i=1}^{n_s+1}\sum_{j=1}^{n_s+1}\sum_{k=1}^{j}\sum_{l=1}^{j-k+1} \beta_{k-1}^{j-1}\beta_{l-1}^{j-k}\bar{B}_{j-k-l+1}^T P_{ji}\Lambda_{ik}\bar{C}_{l-1}^T + 2\sum_{i=1}^{n_s+1}\sum_{j=1}^{n_s+1}\sum_{k=\max(i,j)}^{n_s+1} \beta_{j-1}^{k-1}\beta_{i-1}^{k-1} R_{k-i} L\bar{C}_{k-j}\Lambda_{ji}\bar{C}_{k-i}^T = 0;$$

where $P_{ij} \in R^{n \times n}$ is the $(i,j)^{th}$ sub-matrix of P, $\Lambda_{ij} \in R^{n \times n}$ is the $(i,j)^{th}$ sub-matrix of Λ,

$$A = \begin{bmatrix} \tilde{A} & 0 & 0 & \dots & 0 \\ \tilde{A}_1 & \tilde{A} & 0 & \dots & 0 \\ \tilde{A}_2 & 2\tilde{A}_1 & \tilde{A} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{A}_{n_s} & \beta_1^{n_s}\tilde{A}_{n_s-1} & \beta_2^{n_s}\tilde{A}_{n_s-2} & \dots & \tilde{A} \end{bmatrix}, D^T = \left[\bar{D}^T, \bar{D}_1^T, \bar{D}_2^T, ..., \bar{D}_{n_s}^T\right], \tilde{A} = \bar{A} + \bar{B}L\bar{C},$$

$$\tilde{A}_i = \bar{A}_i + \sum_{j=1}^{i+1} \beta_{j-1}^i \bar{B}_{i+1-j} L\bar{C}_{j-1}, \bar{A}_i = \frac{\partial^i \bar{A}}{\partial \alpha^i}, \bar{B}_i = \frac{\partial^i \bar{B}}{\partial \alpha^i}, \bar{C}_i = \frac{\partial^i \bar{C}}{\partial \alpha^i}, \bar{D}_i = \frac{\partial^i \bar{D}}{\partial \alpha^i}, \beta_j^i \equiv \frac{i!}{j!(i-j)!}, Q_{ij} \in R^{n \times n},$$

the $(i,j)^{th}$ sub-matrix of Q is given by $Q_{ij} = \delta_{ij}\bar{Q}_{i-1} + \sum_{k=\max(i,j)}^{n_s+1} \beta_{i-1}^{k-1}\beta_{j-1}^{k-1}\bar{C}_{k-i}^T L^T R_{k-1} L\bar{C}_{k-j}$; $\delta_{ij} = \begin{cases} 1 & i=j \\ 0 & i \neq j \end{cases}$; and all matrices are evaluated at the nominal value $\alpha = \alpha_0$.

Future work will address the following questions: (i) Does this approach lead to controllers that have superior stability and performance robustness properties when compared to the standard LQ design? and (ii) Can the results be easily extended to multiple parameters?

ACKNOWLEDGEMENTS

The support of this research through a grant from the Ohio University Stocker Faculty Research Fellowship and the National Science Foundation (MSS-9111080) is hereby acknowledged.

REFERENCES

1. Yedavalli. R. K. and R. E. Skelton. 'Controller Design for Parameter Sensitivity Reduction in Linear Regulators'. *Optimal Control Applications & Methods*. **3**. 221-240 (1982).
2. Okada. K. and R. E. Skelton, 'Sensitivity Controllers for Uncertain Systems', *J. Guidance*, **13**, 321-329 (1990).

INTERNAL MODELS IN TWO-DEGREES-OF-FREEDOM CONTROL SYSTEMS

Oscar R. González[1]
Department of Electrical and Computer Engineering
Old Dominion University
Norfolk, VA 23529-0246
org100f@eefs01.ee.odu.edu

INTRODUCTION

Conditions for a two-degrees-of-freedom (2DOF) controller to solve the robust tracking and disturbance rejection problem have appeared in the literature (eg. see [1-5]). In general, they consist of two conditions. One condition, is the well known requirement for C_y to have an internal model as in 1DOF control systems. The second condition is in terms of the controller and plant parameters. The contribution of this paper is to clarify the role of this latter condition and to show two particular 2DOF controller implementations for which this latter condition is robustly satisfied. In this way, the inclusion of a desired internal model using a systematic design technique such as H_∞ or mu-synthesis will be easier.

ROBUST TRACKING, INTERNAL MODELS, AND 2 DOF CONTROLLERS

Consider the control system shown in Figure 1. The plant is described by $[y'_c \; y'_m]' = P\,[u' \quad w']'$ and $P = P_{ij}$ $i,j = 1,2$ are proper real rational transfer matrices of appropriate dimensions, ie. $P_{ij} \in M(\mathbb{R}_p(s))$. It is assumed that the subsystem with I/O map $y_m = P_{22}u$ is stabilizable and detectable. In addition, let $y_c = Q_m y_m$ with Q_m a stable rational transfer function (not necessarily proper) with transmission zeros disjoint from the poles of T_{rv}. This is a general condition given in [8] for which the disturbance rejection and tracking problems are solvable. An alternate mode readability condition is given in [5]. The 2DOF controller is described by $u = C\,[y'_m \; u']'$, $C = [-C_y \;\; C_r] = \tilde{D}_c^{-1}\left[-\tilde{N}_y \quad \tilde{N}_r\right]$ a left coprime RH_∞ factorization. The request input vector r is characterized by $r = T_{rv}v$, v an unknown bounded vector and T_{rv} antistable. The input/output mapping of the control system is denoted by $y_c = T_{cr}r$.

The asymptotic tracking problem with internal stability is to find a stabilizing controller C such that $(I - T_{cr})T_{rv} \in M(RH_\infty)$ iff $(I - P_{12}DX)\tilde{D}_r \in M(RH_\infty)$, where $P_{22} = ND^{-1}$, $T_{rv} = \tilde{D}_r^{-1}\tilde{N}_r$ are proper and stable coprime factorizations and $P_{12}D,\, X \in M(RH_\infty)$. Robust tracking means that the above nominal performance condition and internal stability are maintained under plant parameter variations and a subset of controller parameter variations. When $y_c = Q_m y_m$, a set of robust tracking conditions can be written using [5].

Theorem. Suppose the nominal closed-loop system is internally stable. Then, C is a robust tracking compensator if and only if

(i) $(I - Q_m NX)\tilde{D}_r^{-1} \in M(RH_\infty)$, and

(ii) $Q_m D_{C_y}/\alpha_r \in M(RH_\infty)$

where $C_y = N_{C_y}(D_{C_y})^{-1}$ is a proper and stable right coprime factorization and α_r is the largest

---

[1] Partially supported by NASA Grant NAS1-18584-133.

invariant factor of $\tilde{D}_r$.

A similar set of conditions for robust tracking when $y_c = y_m$ is given in [1, Theorem 7.5.37] where the second condition remains the same and the first condition is shown to be equivalent to $(\tilde{N}_r - \tilde{N}_y)\tilde{D}_r^{-1} \in M(RH_\infty)$ when P is square.

Both of these theorems for 2DOF control systems consider only perturbations in the plant and none in the controller. They include an internal model of T_{rv} in C_y. 1DOF control systems have the additional property that condition (ii) implies condition (i). In addition, robust tracking is maintained under plant parameter variations and parameter variations in part of the 2DOF controller that do not affect the internal model and internal stability. In general, 2DOF control systems require both conditions in the theorem to be independently satisfied. Particular implementations of 2DOF controllers, however, do satisfy the 1DOF property. When $y_c=y_m$, Grimble has referred to one such implementation as a 2 ½ DOF controller (see eg. [10]). This 2 ½ DOF controller is shown in Figure 2. If C_r is unstable, then the controller cannot, in general, be realized in this manner. One possible implementation is shown in Figure 3.

Notice that this is one more result that shows that not all 2DOF controller implementations may be suitable (see [7]). It would be useful to characterize those 2DOF controller implementations that satisfy a similar property to 1DOF control. One approach is to consider the results in [6], which give a way to characterize a set of maps that contain an internal model in different control problems. In particular, for robust tracking, when $y_c=y_m$, it is shown that the internal model must appear in

$$\hat{C} = (I + C_y P_{22})^{-1} C_r [I + P_{22}(C_y - C_r)]^{-1} (I + P_{22} C_y)$$

For example, if $C_y=C_r$, then $\hat{C} = C_y = C_r$ as expected. Consider now the 2DOF controller in Figure 4. This implementation was selected in [12] as preferable when compared to five other 2DOF controllers. Substituting C_y and C_r in $\hat{C}$, shows that if an internal model of T_{rv} is present in both G and G_o, then $\hat{C}/\alpha_r \in M(RH_\infty)$ and condition (i) of the theorem is satisfied. For this implementation, two internal models are required, one inside the main feedback loop and one inside the feedback loop comprising C_r. Further research is needed to understand the role of 2DOF controller realizations in control system design.

CONCLUSIONS

When $y_c=y_m$, it has been shown that for particular 2DOF controller implementations robust tracking requires the presence of an internal model in C_y and C_r appropriately. This allows plant parameter variations of a specified type and variations in the controller parameter as long as internal stability and the internal model are maintained. In general, for 2DOF controllers additional restrictions in the controller parameters are required. If $y_c=Q_m y_m$, then the 1DOF property is not satisfied; the first condition of the theorem needs to be satisfied in addition to the presence of an internal model. Of course, the implementation of robust tracking controllers represents a rather conservative approach. A systematic design technique such as H_∞ or mu-synthesis could be used to minimize a frequency weighted measure of the error and other signals (see eg. [9-11]). The design approaches in [9-11], however, consider particular 2DOF controllers. For control system design, it is better not to use a particular implementation during the design, but rather to leave the implementation for a later step.

REFERENCES

[1] Vidyasagar, M. 1985, *Control System Synthesis: A Factorization Approach*, (Massachusetts: MIT Press).
[2] Desoer, C.A. and Gustafson, C.L., 1984, *I.E.E.E. Transactions on Automatic Control, 29*, 909-917.
[3] Sugie, T. and Yoshikawa, T., 1986, *I.E.E.E. Transactions on Automatic Control, 31*, 552-554.
[4] Hara, S. and Sugie, T., 1988, *I.E.E.E. Transactions on Automatic Control, 33*, 59-67.
[5] Sugie, T. and Vidyasagar, M., 1989, *Systems and Control Letters, 13*, 101-108.
[6] Gonzalez, O.R. and Antsaklis, P.J., 1991, *International Journal of Control, 53*, 411-430.
[7] Gonzalez, O.R. and Antsaklis, P.J., 1989, *Proceedings of the 1989 American Control Conference*, 269-272.
[8] Gonzalez, O.R. and Antsaklis, P.J., 1989, *I.E.E.E. Transactions on Automatic Control, 34*, 885-888.
[9] Yaesh, I. and Shaked, U., 1991, *I.E.E.E. Transactions on Automatic Control*, 36, 1272-1276.
[10] Grimble, M.J., 1992, *IEE Proceedings-D, 139*, 147-160.
[11] Hoyle, D.J., Hyde, R.A., and Limebeer, D.J.N., 1991, *Proceedings of the 30th CDC*, 1581-1585.
[12] Desoer, C.A. and Lin, C.A., 1985, *International Journal of Systems Science, 16*, 789-813.

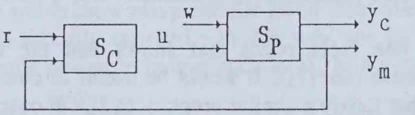


Figure 1. The compensated system $\Sigma(S_P,S_C)$.

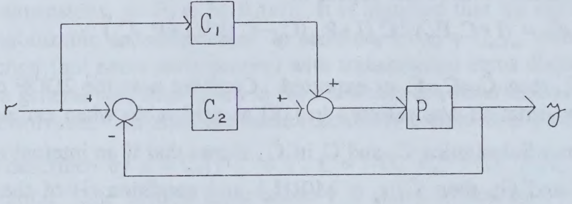


Figure 2. 2 1/2 DOF Control System

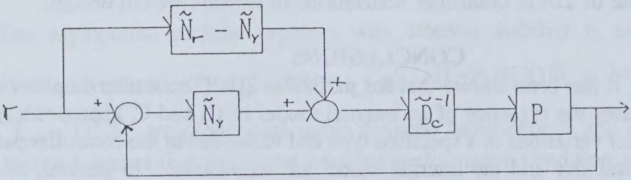


Figure 3. An Implementation of a 2 1/2 DOF Control System

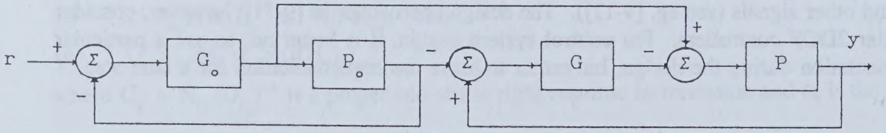


Figure 4. 2 DOF Control System

BLOCK DIAGONAL OUTPUT COVARIANCE CONSTRAINT PROBLEM FOR PERIODIC AND MULTIRATE SYSTEMS

Guoming Zhu and Robert E. Skelton
Space Systems Control Lab, Purdue University, W. Lafayette, IN

1 Introduction

In practical engineering problems multirate control is sometimes necessary because of limited computational speed and accuracy of real time control computers and a wide range of the controlled system dynamics. If the ratio of any two sampling rates is rational, A continuous system sampled at multiple rates can be described with a periodically time-varying model, see [1]. This paper presents an algorithm to choose the weight matrices in the LQG problem by defining a physically meaningful set of specifications. By solving the discrete periodic Block Output Covariance Constraint (BOCC) problem, the output weighting matrix Q in LQG problem will be chosen to satisfy the output covariance constraints.

2. The BOCC Problems for Discrete Periodic Systems

Suppose that the continuous system is sampled at μ different rates $1/T_1, 1/T_2, ..., 1/T_\mu$. If the ratio of any two sampling rates is rational, there exist μ unique integers $p_1, p_2, ..., p_\mu$ and a unique real number Δ such that $T_i = p_i\Delta$, where the lowest common multiple of $p_1, p_2, ..., p_\mu$ is equal to one [2]. Then the continuous system can be discretized into the periodically time-varying system described below with sampling rate $1/\Delta$.

$$\left.\begin{aligned} x_p(k,n+1) &= A_p(n)x_p(k,n) + B_p(n)u(k,n) + D_p(n)w_p(k,n) \\ y_p(k,n) &= C_p(n)x_p(k,n) \ ; \ z(k,n) = M_p x_p(k,n) \end{aligned}\right\} ; \ n = 0, 1, ..., T_p-1 , \tag{1}$$

where T_p is the period of the given system (1). Define Δ to be the sampling period of (1), then the pair (k,n) represents the time axis at $(T_p k + n)\Delta$. Vector $w_p(\cdot, \cdot)$ is a zero mean cyclo-stationary process with periodic covariance matrix $W_p(n)$.

Suppose that the outputs of (1) are in the following form $y_p = [y_1^T, y_2^T, ..., y_m^T]^T$, where $y_i^T \triangleq [y_1^i, y_2^i, ..., y_{m_i}^i]^T$ $(i = 1, 2, ..., m)$. Define the periodic output covariance matrix for each output vector $y_i(\cdot,n)$ as $Y_i(n) \triangleq \lim_{k\to\infty} \mathcal{E}\, y_i(k,n)y_i^T(k,n)$. Suppose that the closed loop system (1) can be written into the following form

$$x(k,n+1) = A(n)x(k,n) + D(n)w(k,n) \ ; \ y(k,n) = C(n)x(k,n) \ , \tag{2}$$

where $C(n) \triangleq [C_y^T \ C_u^T]^T$, $w = [w_p^T \ v^T]^T$, $y = [y_p^T \ u^T]^T$ and $u = C_u x$. Then the stochastic BOCC problem for periodic discrete systems can be defined as follows.

The BOCC Problem for Periodic Systems

Find a periodic static or fixed order dynamic controller to minimize the BOCC cost

$$J_{BOCC} = \sum_{n=0}^{T_p-1} \operatorname{tr} R(n)C_u(n)X(n)C_u^T(n) \ ; \ R(n) > 0 \tag{3}$$

subject to

$$X(n+1) = A(n)X(n)A^T(n) + D(n)W(n)D^T(n) \tag{4}$$

$$Y_i(n) \le \bar{Y}_i(n) \ ; \ \bar{Y}_i(n) > 0 \ ; \ i = 1, 2, ..., m \ ; n = 0, 1, ..., T_p-1 , \tag{5}$$

where $Y_i(n) = C_i(n)X(n)C_i^T(n)$ and $C_y(n) \triangleq [C_1^T(n), C_2^T(n), ..., C_m^T(n)]^T$.

3. Necessary Conditions and Computational Algorithm for Periodic BOCC Problems

Consider the stochastic BOCC problem. Let $w_p(k,n)$ be a zero mean cyclo-stationary process with periodic static covariance matrix $W(n)$ defined by

$$\lim_{k\to\infty} \mathcal{E}\,[w_p(k_1,n_1)w_p^T(k,n)] = W_p(n)\delta(k_1 - k)\delta(n_1 - n). \tag{6}$$

Firstly, we consider the measurement feedback case. Let the periodic time-varying controller be $u(k,n) = G(n)z(k,n)$; $z(k,n) = M_p(n)x_p(k,n)$. In this case the closed loop system matrices in (2) are defined by

$$A(n) = A_p(n) + B_p(n)G(n)M_p(n) \;;\; D(n) = D_p(n) \;;\; C_y(u) = C_p(n) \;;\; C_u(n) = G(n)M_p(n) \quad .(7)$$

Theorem 1

Suppose that the measurement feedback gain G(n) is the optimal solution of the periodic BOCC problem. Then there exists a $Q(n) = \mathrm{diag}[Q_1(n), Q_2(n), \ldots, Q_m(n)] \geq 0$; $Q_i \in \mathbb{R}^{m_i \times m_i}$; $i = 1, 2, \ldots, m$ for $n = 0, 1, \ldots, T_p-1$ such that

$$[R(n)+B_p^T(n)K(n+1)B_p(n)]G(n)M_p(n)X(n)M_p^T(n) = -B_p^T(n)K(n+1)A_p(n)X(n)M_p^T(n) \;; \tag{8}$$

$$K(n) = A^T(n)K(n+1)A(n) + M_p(n)^T G^T(n)R(n)G(n)M_p(n) + C_p^T(n)Q(n)C_p \;;(n) \tag{9}$$

$$X(n+1) = A(n)X(n)A^T(n) + D_p(n)W_p(n)D^T(n) \;; \tag{10}$$

$$0 = [C_i(n)X(n)C_i^T(n)-\bar{Y}_i(n)]Q_i(n) \;,\; i = 1, 2, \ldots, m \;;\; n = 0, 1, \ldots, T_p-1 \;. \tag{11}$$

Note that the BOCC controller is a quadratic optimal one with a special choice of the output weighting matrix Q. To solve the BOCC problem, an computational algorithm is developed. The BOCC Algorithm is very much similar to the continuous BOCC algorithm in [2,3]. Hence, we only give an outline.

1) Given the known parameters and matrices.

2) Solve (8-11) for G(n) and X(n) with a fixed Q(n).

3) If $\sum_{n=0}^{T_p-1} \sum_{i=1}^{m} \| Q_i(n)[C_i(n)X(n)C_i(n) - \bar{Y}_i(n)] \| < \varepsilon$, stop. Else,

$$Q_i(n) = \bar{Y}_i^{-\alpha}(n)Y_i^{\alpha}(n)Q_i(n)Y_i^{\alpha}(n)\bar{Y}_i^{-\alpha}(n) \;;\; Q(n) = \mathrm{diag}[Q_1(n), Q_2(n), \ldots, Q_m(n)] \tag{12}$$

for $n = 0, 1, \ldots, T_p-1$, and return to 2).

Note that if the periodic BOCC algorithm converges the BOCC algorithm produces a feasible solution of the BOCC problem. This fact is proved in [3].

For multiorder-multirate control we consider a dynamic system sample at two different rates 1/T and 1/Δ. Suppose that T/Δ is an integer. Then, one can put the system into the standard periodically time-varying form (1) with $T_p = T/\Delta$ such that vector u_1 and z_1 are sampled at 1/T, and u_2 and z_2 at 1/Δ, where $u \triangleq [u_1^T \; u_2^T]^T$; $z \triangleq [z_1^T \; z_2^T]^T$. We are going to design the multirate controller in the following form

$$\left.\begin{aligned} x_c(k,n+1) &= [P(n)A_c(n)P(n) + I - P(n)]x_c(k,n) + P(n)B_c(n)z(k,n) \\ u(k,n) &= C_c(n)P(n)x_c(k,n) + D(n)z(k,n) \end{aligned}\right\} . \tag{13}$$

where $P(0) = I_{nc}$; $P(n) = \text{block diag}[0_{nc-nc_1} \; I_{nc_1}]$; $n = 1, 2, \ldots, T_p-1$, and suffixes nc and nc_1 are the controller order for the slow and fast rate 1/T and 1/Δ. Partition matrices A_c, B_c, C_c and D_c into 2×2 blocks, and note that $M_p(n) = [0 \; M_{p_2}^T]^T$; $B_p(n) = [0 \; B_{p_2}(n)]$; $n = 1, 2, \ldots, T_p-1$. Then for n=0 the controller has order nc, and for n≠0 the controller has order nc_1. Since (12) is in the form of measurement feedback, the BOCC algorithm can be applied to design the multiorder multirate controller (13).

4. Conclusion

The Block diagonal Output Covariance Constraint problem for the discrete periodic system is defined, and the derived necessary conditions of the problem for both measurement and full order dynamic feedback are solved by a feasible iterative algorithm (called the BOCC algorithm). It turns out that the resulting controller is a periodic LQG controller with a special choice of the output weight matrices (periodic). The BOCC algorithm can be also applied to design multiorder multirate controllers.

References

[1] R. E. Kalman and J. Bertram, "A Unified Approach to the Theory of Sampling Systems," *J. Franklin Inst., Vol. 267, 1959.*

[2] G. Zhu, "$\mathcal{L}_2$ and $\mathcal{L}_\infty$ Multiobjective Control for Linear Systems," *Ph.D. Dissertation, Purdue University, May, 1992.*

[3] G. Zhu and R. Skelton, "Controller Design to Achieve Covariance Constraints," *IFAC Symposium on Design Methods of Control System, Zurich, Switzerland,Sep., 1991.*

Bidirectional Distributed Deadlock Detection and Resolution Algorithm

Yongtae Shin and Don Epley
Department of Computer Science
The University of Iowa
Iowa City, IA 52242
shin@herky.cs.uiowa.edu
epley@herky.cs.uiowa.edu

ABSTRACT

We propose a new algorithm for a distributed deadlock detection and its resolution. The algorithm consists of two phases, the deadlock detection phase and the deadlock resolution phase. The algorithm initiates deadlock detection messages that are sent along the edges of WFG bidirectionally, and detects the deadlock when the collision of the messages occurs. The resolution phase resolves a deadlock by the abortion of a lowest priority process that is involved in the deadlock. The algorithm is stated in terms of processes, and we employ the UNITY approach for the proof of correctness.

I. INTRODUCTION

Deadlock detection and resolution are an important issue of resource management in distributed systems since resource allocation may result in deadlock, and it is closely associated with concurrency. Since deadlock is a real problem that can cause expensive, time-dependent hang-ups or failures in distributed systems, its handling is one of the most important research and development problems.

A first probe based distributed deadlock detection algorithm designed in [Chandy 82]. [Chandy 82] introduced the probe computations that pass a message called *probe* from process to process. Later, [Sinha 85] introduced the priority based probe algorithm. [Sinha 85] assigns a unique priority to a transaction and used the priority to reduce the number of probe messages that are propagated. [Choud 89] modified the algorithm of [Sinha 85] which is incorrect and fails to detect all deadlocks, and may report false deadlocks. However, the problems have been reported in [Choud 89] as well by [Kshem 91]. [Kshem 91] eliminated deficiencies of [Choud 89], and formally proved the correctness of their algorithm based on invariant method. All deadlock detection algorithms we discussed above is based on the edge-chasing algorithms in which probes are sent upon request and along the outgoing edge of the Wait-for-Graph (WFG). [Mitch 84] developed an algorithm for the deadlock detection which is also based on the edge-chasing algorithm but probes are sent upon request and in the opposite direction of edges in the WFG.

Since a deadlock is the time dependent state of permanent blocking of a set of processes, it is important to detect as soon as possible and resolve it. Our algorithm is a first attempt to send the probe-type messages bidirectionally, i.e., sending the probe type messages in both directions of the edge in the WFG, to detect a deadlock in the system.

Unlike other priority based algorithms, we adapt a process model referred to in [Herma 83]. Hence, our discussion is in terms of processes.

Propagating the probe type messages in both incoming and outgoing directions reduces an elapsed time of detecting the deadlock. The *detection time complexity* is time elapsed from the initiation of the algorithm until the detection of the deadlock, given that message transmission delay and inter-message delay have a certain upper bound constants. Since our algorithm detects a deadlock only by the highest priority process, in best case the reduction of the detection time complexity can be half of other edge-chasing algorithms.

We present the algorithm in UNITY formalism. Since many deadlock detection and resolution algorithms used informal or intuitive reasoning to prove their correctness, most of them are reported incorrect. Hence we employ the UNITY formalism, and avoid the operational reasoning as much as possible to prove safety and progress properties of the algorithm.

Our paper is organized as follows: Section II contains a model of distributed systems. The first part of this section presents the notations used for the discussion, and the remain of the section describes about the system model. Section III contains both informal and formal descriptions of the algorithm. The readers may refer to [Shin 92] for the correctness proofs. Finally, concluding remarks are presented in section IV.

II. A MODEL OF DISTRIBUTED SYSTEMS

II.1. Preliminaries

A distributed system is a collection of processes connected by a communication network. Processes are reactive programs that are capable of exchanging messages with one another. We assume that if process u can send messages to process v, it can also receive messages from v. A process u has an unique identification number $id(u)$, and for simplicity, we use its identification number for its priority assigned. We assume a priority is assigned to a process using the timestamp mechanism that is used in [Sinha 85]. For two processes u and v, $u > v$ if and only if $id(u) < id(v)$.

Furthermore, we assume that messages are delivered by a communication channel between the processes in a finite time but within a certain upper bound K, with error-free, and the order they sent (FIFO). Two messages arriving in the same time are taken by an arbitrary order at the receiver. A receiver can receive a message only if the channel has a message, and a message is removed from the channel after it is received. A channel is represented by a sequence of messages. We use notation $chnl_{u,v}$ for a communication channel from process u to process v. For a channel c, hd(c) and tl(c) denotes head and tail of c, respectively. The sending of message msg from u to v is equivalent to the assignment

$$chnl_{u,v} := chnl_{u,v}@msg$$

where @ denotes adding msg into the end of $chnl_{u,v}$.

The receiving of the message from u at v is equivalent to the assignment

$$var_v := \mathrm{hd}(chnl_{u,v})$$

where var_v is a local variable in v that can accommodate the message. The removing of the message from the channel c is denoted by tl(c) operation.

In general, a message consists of several *fields*. The following notation will be used to assign a *field* of a message at the head of the channel to a local variable:

$$var_v := \mathrm{hd}(chnl_{u,v}).field.$$

A notation $\perp$ is used for an undefined type of message.

We can abstract the state of a distributed system by a wait-for graph (WFG) [Holt 72]. A deadlock is indicated by a cycle in the WFG. Although a WFG is not explicitly constructed by our algorithm, it will be referred to simplify our discussion. There is one-to-one correspondence between vertices in a WFG and processes in the system. A direct edge (u, v) forms when a process u blocks on process v. Process u is called the *requester* and process v is called the *holder*. The WFG changes due to various actions in the system. The detail description of these changes is given in section II.2.3 to help us to understand the formation and resolution of a distributed deadlock.

This paper deals with a single request model[1]. In this model a process can have at most one outstanding resource request at a time. This implies that the maximum outdegree of the relevant WFG is 1. To find deadlocks in a system which is modeled by the single-request model, it is necessary to find a cycle in the WFG. The single-request model can be described as a forest of isolated trees in the WFG if no deadlock appears yet in the system. Suppose T be a set of all processes in the system. Then we can decompose T as $\mathrm{T} = \mathrm{T}(1) \cup \mathrm{T}(2) \cup \ldots \cup \mathrm{T}(n)$, where $\langle \forall i,j : i \neq j,\ 1 \leq i,j \leq n :: \mathrm{T}(i) \cap \mathrm{T}(j) = \varnothing \rangle$. The index of T's is the root node of T. Whenever the root of any tree $\mathrm{T}(i)$ begins waiting on one of its descendants, the $\mathrm{T}(i)$ becomes nontree $\mathrm{NT}(i)$, and $\mathrm{NT}(i)$ is deadlocked. At any instant $\mathrm{NT}(i)$ can contain only a single deadlock, and all the processes in the $\mathrm{NT}(i)$ are blocked.

The spontaneous aborts of processes due to the deadlock resolutions are excluded. Hence, a deadlock is a stable property until it is detected and resolved. A deadlock is no longer stable property of a distributed system if either multiple request or spontaneous aborts are allowed. A process can be part of several deadlocks concurrently in the multiple request model. When a process detects a deadlock the existence of a deadlock at that instant is not guaranteed since a deadlock process might have spontaneously aborted or might have been aborted by a victim of other deadlocked processes.

II.2 System Model

II.2.1. The Messages[2]

There are two deadlock detection messages, *in* and *out*. Both *in* and *out* are ordered pairs (*initiator*, *junior*) where *junior* represents the lowest priority process that either *in* or *out* has traversed. We also employ two resolution messages, *clean_in* and *clean_out*. The *clean_in* and *clean_out* messages are also an ordered pair (*victim*, *initiator*) where *victim* is a *junior* process when a deadlock is detected by the *initiator*.

We define the following axioms for the behavior of the messages.

A1. (behavior of *in* message):

$$\langle \forall u, v : u \text{ and } v \text{ are process id} :: chnl_{u,v}@in(initiator, junior) \Rightarrow (u \in \mathrm{HOLD}_v \wedge v \in \mathrm{REQ}_u \wedge initiator > v)\rangle$$

A2. (behavior of *out* message):

$$\langle \forall u, v : u \text{ and } v \text{ are process id} :: chnl_{u,v}@out(initiator, junior) \Rightarrow (u \in \mathrm{REQ}_v \wedge v \in \mathrm{HOLD}_u \wedge initiator > v)\rangle$$

A3. (behavior of *clean_in* message):

$$\langle \forall u, v : u \text{ and } v \text{ are process id} :: chnl_{u,v}@clean\_in(victim, initiator) \Rightarrow (u \in \mathrm{REQ}_v \wedge v \in \mathrm{HOLD}_u \wedge u \neq initiator)\rangle$$

---

[1] Our approach can be extended to a multiple-request model.

[2] We exclude the discussion of primitive messages such as *request* and *grant* for the purpose of simplicity.

A4. (behavior of *clean_out* message):

$$\langle(\forall u, v : u \text{ and } v \text{ are process id} :: chnl_{u,v}@clean\_out(victim, initiator)$$
$$\Rightarrow (u \in \text{HOLD}_v \wedge v \in \text{REQ}_u \wedge u \neq initiator)\rangle$$

Both *in* and *clean_out* messages propagates along the incoming edges while messages *out* and *clean_in* propagates along the outgoing edge in the WFG. **A1** and **A2** enforce only deadlock messages initiated by the highest priority process traverse through a deadlock cycle. The axioms reduce the number messages generated by the algorithm.

Other than the deadlock and resolution messages, we have messages called *abort* and *signal*. Message *abort* is an ordered pair (u, i) where u is a process which detects a deadlock and i is an initiator of the deadlock algorithm. A detector u of deadlock sends an *abort*$(u, initiator)$ to a victim after it detect a collision of *in* and *out*. A victim process which receives an *abort* starts the resolution phase. A *signal* message is used to notice the completion of updating the INQs and OUTQs of processes that are involved in a deadlock cycle.

II.2.2. The Local Variables

Each process u has local variables REQ_u and HOLD_u. REQ_u is a set of processes that have requested for the resource which is held by u, and HOLD_u is set of a process that u is waiting on[3]. Moreover, a process u maintains two sets, called INQ_u and OUTQ_u, which contain *in* and *out* messages received, respectively. The INQ_u contains information relating to processes which caused u to be blocked either directly of transitively whereas the OUTQ_u contains information relating to processes which are waiting on u either directly or transitively. We use notation "$\cup$" and "$\backslash$" for an insertion and a deletion of message in INQ_u and OUTQ_u, respectively. For example, $\text{INQ}_u \cup (i, j)$ denotes the insertion of $in(i, j)$ into INQ_u, and $\text{INQ}_u \backslash (i, j)$ denotes the deletion of $in(i, j)$ from INQ_u. We assume that there is no duplicate message in the sets.

A process u can be in either one of four states, denoted by STATE_u: *active*, *waiting*, *pending*, and *aborting*. In terms of the WFG a root of tree process is in active state, and the leaves of tree are in waiting state. The rest of processes which are the internal processes of a tree are in pending state. A process enters into aborting state when it receives *abort* message from the process which detected a deadlock. Only a root process which is in active state can request for a resource, and all other processes are blocked by its holder. We assume that a process does not request a resource to itself. A process goes into active state from either waiting or pending state when it issues a request to another process for the resource. A process that is in either waiting or pending state enters into active state when it receives a requested resource from the holder, and we call this process a *newholder* of the resource.

II.2.3. Behavior of the System

We describe changes in the WFG due to various actions in the system, and we assume these actions are performed atomically and embedded in our algorithm. Let a tree *r.tree* and a nontree *r.nontree* to be a set of processes of tree rooted at r and nontree rooted at s, respectively. Only roots of trees and aborting processes can make changes in WFG, i.e., either by the creation of edge or by the deletion of edge. When process u waits on process v an edge (u, v) forms. When the root process r of *r.tree* waits on a node in *r.tree*, the *r.tree* becomes to nontree *r.nontree*. A edge (u, v) is removed when process v grants a resource or either *u* or *v* aborts. We define following rules to help understanding the formation and resolution of the system. The notation "x; y" indicates sequential execution of x and y.

---

[3] Note that REQ is a set of processes while HOLD is a set of a single process since our deadlock model deals with a single-request model.

R1. When process $r \in r.tree$ requests a resource to process $q \in r.tree$, an edge(r, q) forms:
$\mathrm{HOLD}_r, \mathrm{REQ}_q, r.nontree := \{q\} \cup \mathrm{HOLD}_r, \{r\} \cup \mathrm{REQ}_q, r.tree;$
$r.tree, \mathrm{STATE}_r := \varnothing, pending;$
$\langle \forall (i, j) : (i, j) \in \mathrm{OUTQ}_r :: chnl_{r,q} := chnl_{r,q}@out(i,j) \rangle;$
$\langle \forall (i, j) : (i, j) \in \mathrm{INQ}_q :: chnl_{q,r} := chnl_{q,r}@in(i,j) \rangle$

R2. When process $r \in r.tree$ requests a resource to process $q \in p.tree$, $r \neq p$, an edge(r, q) forms:
$\mathrm{HOLD}_r, \mathrm{REQ}_q, p.tree := \{q\} \cup \mathrm{HOLD}_r, \{r\} \cup \mathrm{REQ}_q, p.tree \cup r.tree;$
$r.tree := \varnothing;\ \mathrm{STATE}_r := waiting$ **if** $\mathrm{REQ}_r = \varnothing \sim pending$ **if** $\mathrm{REQ}_r \neq \varnothing$
$\langle \forall (i, j) : (i, j) \in \mathrm{OUTQ}_r :: chnl_{r,q} := chnl_{r,q}@out(i,j) \rangle;$
$\langle \forall (i, j) : (i, j) \in \mathrm{INQ}_q :: chnl_{q,r} := chnl_{q,r}@in(i,j) \rangle$

R3. When process $r \in r.tree$ requests a resource to process $q \in s.nontree$, $s \neq r$, an edge(r, q) forms:
$\mathrm{HOLD}_r, \mathrm{REQ}_q, s.nontree := \{q\} \cup \mathrm{HOLD}_r, \{r\} \cup \mathrm{REQ}_q, s.nontree \cup r.tree;$
$r.tree := \varnothing;\ \mathrm{STATE}_r := waiting$ **if** $\mathrm{REQ}_r = \varnothing \sim pending$ **if** $\mathrm{REQ}_r \neq \varnothing$
$\langle \forall (i, j) : (i, j) \in \mathrm{OUTQ}_r :: chnl_{r,q} := chnl_{r,q}@out(i,j) \rangle;$
$\langle \forall (i, j) : (i, j) \in \mathrm{INQ}_q :: chnl_{q,r} := chnl_{q,r}@in(i,j) \rangle$

It is important to maintain the messages in INQ and OUTQ properly since the misleading information in these sets can cause a detection of false deadlock The update of INQ and OUTQ have to be performed not only in the resolution phase but also whenever a process releases its resource to any of its requester. When a process releases its resource to the *newholder*, it removes all the *out* messages in its OUTQ. Also, the *newholder* have to remove all *in* messages that it sent from the INQ's of its descendants. We can remedy this problem by defining **R4** and **R5**.

R4. When process $r \in r.tree$ releases a resource to process $q \in \mathrm{REQ}_r$, an edge(r, q) removes and $\langle \forall i : i \in \mathrm{REQ}_r - \{q\} ::$ edge(i, q) is formed $\wedge$ edge(i, r) is removed $\wedge$ i propagates *out* messages in OUTQ_i to $q.\rangle$:
$\mathrm{HOLD}_q, \mathrm{REQ}_r, q.tree := \varnothing, \mathrm{REQ}_r - \{q\}, r.tree - \{r\};$
$r.tree, \mathrm{REQ}_q := \varnothing, \mathrm{REQ}_q \cup \mathrm{REQ}_r;$
$\langle \forall i, (j, k) : i \in \mathrm{REQ}_v \wedge (j, k) \in \mathrm{OUTQ}_i ::$
$\quad \mathrm{HOLD}_i, chnl_{i,q} := \{q\}, chnl_{i,q}@out(j, k) \rangle;$
$\mathrm{STATE}_q, \mathrm{OUTQ}_r := active, \varnothing;$
$\langle \forall j : out(j, junior) \in \mathrm{OUTQ}_q \vee j \in \mathrm{REQ}_q :: chnl_{q,j}@\ remove(q) \rangle$

Note that process *q* becomes not only a *newholder* but it becomes a root of a new tree, and any policy can be used to select a *newholder*.

R5. When process $p \in q.tree$ receives *remove*(q) message from *newholder* q, it performs following:
$\mathrm{INQ}_p := \mathrm{INQ}_p \setminus \{(q, junior)\};$
$\langle \forall i : i \in \mathrm{REQ}_p :: chnl_{q,i} := chnl_{q,i}@remove(q) \rangle$

Note that *remove* message is generated by the *newholder* to its all descendants which are in pending state, and then the *remove* messages are propagated to the leaves of the WFG that are processes in waiting state.

R6. When a victim $v \in r.nontree$ cancels a request to $h \in \mathrm{HOLD}_v$ and releases the resource to process $q \in \mathrm{REQ}_v$, both edge(v, h) and edge(q, v) are removed and $\langle \forall i : i \in \mathrm{REQ}_v - \{q\}$: edge(i, q) is formed $\wedge$ edge(i, v) is removed $\wedge$ i propagates *out* messages in OUTQ_i to $q\rangle$:

$\mathrm{HOLD}_v, \mathrm{REQ}_v, q.tree := \emptyset, \mathrm{REQ}_v - \{q\}, r.nontree - \{v\};$
$r.nontree, \mathrm{REQ}_h, \mathrm{HOLD}_q, \mathrm{REQ}_q := \emptyset, \mathrm{REQ}_h - \{v\}, \emptyset, \mathrm{REQ}_q \cup \mathrm{REQ}_v;$
$\langle \forall i, (j, k) : i \in \mathrm{REQ}_v \wedge (j, k) \in \mathrm{OUTQ}_i ::$
$\quad \mathrm{HOLD}_i, chnl_{i,q} := \{q\}, chnl_{i,q}@out(j, k)\rangle;$
$\mathrm{STATE}_v, \mathrm{STATE}_h, \mathrm{STATE}_q := abort, waiting, active$

Note that process q becomes a root of new tree $q.tree$.

III. DEADLOCK DETECTION/RESOLUTION ALGORITHM

The deadlock detection/resolution algorithm is presented in UNITY notion [Chandy 88]. The algorithm is proven by showing that the program satisfies both *safety* and *progress* properties. The safety property describes that if the deadlock is detected by the algorithm, it must indeed exist. On the other hand, the progress property indicates that every deadlock must be eventually detected by the algorithm. We omit the proof of correctness in this version, and readers may refer to [Shin 92] for complete description of the proof.

The algorithm has two phases: the deadlock detection phase and deadlock resolution phase. In deadlock detection phase, process j initiates a deadlock detection if the following condition holds:

$$\langle \exists i,k : i \in \mathrm{REQ}_j \wedge k \in \mathrm{HOLD}_j :: j > k \wedge i < j\rangle.$$

We call the process j an *initiator* of the deadlock detection algorithm. We established the stronger condition for initiation of algorithm than any other priority based algorithms to reduce the number of messages used.

We define the predicates $decreasing\_path_{u,v}$ and $increasing\_path_{u,v}$ on process pair u and v as :

$decreasing\_path_{u,v} \equiv$ there is a path from u to v in the WFG such that all processes k (including v) in the path holds $u > k$, i.e., u is a highest process in that path and *all edges in the path from u to v are outgoing edges.*

$increasing\_path_{u,v} \equiv$ there is a path from v to u in the WFG such that all processes k (including u) in the path holds $v > k$, i.e., v is a highest process in that path and *all edges in the path from u to v are incoming edges.*

Note the fact that both $decreasing\_path_{u,v}$ and $increasing\_path_{u,v}$ have a transitive property.

Pending process u, which holds the initiation condition, initiates the algorithm by sending $in(u, u)$ messages to all processes $r \in \mathrm{REQ}_u$ and $out(u, u)$ messages to a process $h \in \mathrm{HOLD}_u$. The *in* messages are propagated by a holder to its requesters while the *out* message is sent by requester to its holder, i.e., the *in* traverses along the incoming edges while the *out* traverses along an outgoing edge.

When process u receives $in(init, jun)$ from process w, $w \in \mathrm{HOLD}_u$, u propagates *in* messages to all processes $r \in \mathrm{REQ}_u$ if it has not sent an *in* message earlier. u declares deadlock if u already received the *out* message in which the initiator is same as *init*, and sends an *abort* message to the proper victim. Similarly, when process *u* receives *out*(*init*, *jun*) from process *w*, $w \in REQ_u$, *u* propagates *out* messages to a process $r \in HOLD_u$ if it has not sent an *out* message earlier. *u* declares deadlock if *u* already received the *in* message in which the initiator is same as *init*, and sends an *abort* message to the proper *victim*. Hence, a deadlock is detected when the messages *in* and *out* that are generated by a same initiator collides at some process (called a *detector*). Note that the collision of deadlock detection messages occurs only at the highest priority process.

The second phase in which the deadlock is resolved is generated only after a process has detected a deadlock. A detector resolves the deadlock by aborting the lowest priority process, called *victim*, in the deadlock cycle. The detector *u* sends a message *abort*(*u*, *initiator*) to an victim *v*.

When a victim *v* receives a *abort*(*u*, *i*) from *u*, *v* generates the resolution messages, *clean_in*(*v*, *i*) and *clean_out*(*v*, *i*) to update all INQ_u and $OUTQ_u$ in the cycle before it is aborted, where *i* is an initiator of deadlock detection messages which are collided at *u*. *v* discards any message it receives from now on. *v* enters the abort phase only when it receives a *signal* from the detector *u* indicating updates of INQ_u and $OUTQ_u$ are completed. Messages *clean_in*(*v*, *i*) (*clean_out*(*v*, *i*)) travel through the incoming edges (outgoing edge), respectively, until it reach to *i*. Receiving *clean_in*(*v*, *i*) (*clean_out*(*v*, *i*)), all deadlocked processes remove *in* (*out*) messages in INQs (OUTQs) that are sent on behalf of *v* or propagated by *v*. After a victim receives a *signal* from *i* that indicates all the updates of INQ and OUTQ have been accomplished, it enters into the abort phase.

III.2.2. Deadlock Detection and Resolution Algorithm

```
Program Deadlock_Detection_and_Resolution (DDR)

declare
    CONST
        in, out, abort, clean_in, clean_out, signal: integer constant;
    TYPE
        statetype = (active, waiting, pending, aborting);
        ptype = integer;  { type of process id }
        messages = ⟨mtype: integer;
                (init: ptype, jun: ptype)    if mtype = in ∨ out  ~
                (vict: ptype, init: ptype)   if mtype = clean_in ∨ clean_out  ~
                (det: ptype, init: ptype)    if mtype = abort  ~
                (init: ptype)                if mtype = signal  ~
                ⊥       if mtype ≠ (in ∨ out ∨ clean_in ∨ clean_out ∨ abort ∨ signal) ⟩;
    VAR
        chnl: sequence of messages;
        h, i, j, k, r, u, v, *: ptype;        { * denotes an arbitrary process id }
        $detector_u$, $initiator_u$, $junior_u$, $victim_u$ : ptype;
        $INQ_u$, $OUTQ_u$: set of mtype message;
        $REQ_u$, $HOLD_u$: set of ptype;
        $STATE_u$: statetype;
        flag, $initiate_u$, $detect_u$, $rcvdin_u$, $rcvdout_u$,
                $rca_u$, $rci_u$, $rco_u$, $marki_u$, $marko_u$: boolean;
always
I ⟨∀u, v : $STATE_u$ = pending :: $increasing\_path_{u,v}$
```

```
                                  $\Rightarrow in(u, junior) \in \mathrm{INQ}_v \vee \neg\, increasing\_path_{u,v}\rangle$
▮ $\langle \forall u, v : \mathrm{STATE}_u = pending :: decreasing\_path_{u,v}$
                                  $\Rightarrow out(u, junior) \in \mathrm{OUTQ}_v \vee \neg\, decreasing\_path_{u,v}\rangle$
▮ $\langle \forall u, v : \mathrm{STATE}_u = pending :: in(u, junior) \in \mathrm{INQ}_v \Rightarrow increasing\_path_{u,v}\rangle$
▮ $\langle \forall u, v : \mathrm{STATE}_u = pending :: out(u, junior) \in \mathrm{OUTQ}_v \Rightarrow decreasing\_path_{u,v}\rangle$
initially
        $\langle$▮ $u ::$
            $initiate_u, detect_u, rcvdin_u, rcvdout_u, rca_u, rci_u, rco_u, marki_u, marko_u :=$
                      false, false, false, false, false, false, false, false, false
        ||  $\mathrm{INQ}_u, \mathrm{OUT}_u = \varnothing, \varnothing$
        ||  $\langle \mathrm{STATE}_u = active$      if $\mathrm{HOLD}_u = \varnothing$ ~
             $\mathrm{STATE}_u = waiting$     if $\mathrm{HOLD}_u \neq \varnothing \wedge \mathrm{REQ}_u = \varnothing$ ~
             $\mathrm{STATE}_u = pending$     if $\mathrm{HOLD}_u \neq \varnothing \wedge \mathrm{REQ}_u \neq \varnothing$ ~
             $\mathrm{STATE}_u = aborting$    if $\mathrm{hd}(chnl_{u,v}).mtype) = abort \wedge detect_u\rangle$   $\rangle$
assign
$\langle$▮ $u : \mathrm{STATE}_u = pending ::$
    $\langle || r, h : r \in \mathrm{REQ}_u \wedge u > r \wedge h \in \mathrm{HOLD}_u \wedge u > h ::$
        $chnl_{u,r}, chnl_{u,h} := chnl_{u,r}@in(u, u), chnl_{u,h}@out(u, u)$  if $\neg\, initiate_u\rangle$      (S1)
|| $initiate_u :=$ true   $\rangle$                                                                  (S2)
▮
$\langle$▮ $u : \mathrm{hd}(chnl_{w,u}).mtype = in ::$
    $\langle || r : r \in \mathrm{REQ}_u \wedge \neg\, r.flag \wedge r > u \wedge ::$
        $\mathrm{INQ}_u := \mathrm{INQ}_u \cup \{(initiator_u, junior_u)\}$
                if $rcvdin_u \wedge \neg\, detect_u \wedge \neg\, exists(initiator_u, \mathrm{INQ}_u)^4$               (S3)
      || $chnl_{u,r}, r.flag := chnl_{u,r}@in(initiator_u, junior_u)$, true
                    if $rcvdin_u \wedge \neg\, detect_u\rangle$                                         (S4)
    || $detect_u, victim_u :=$ true, $getjunior(initiator_u, \mathrm{OUTQ}_u)^5$
                    if $exists(initiator_u, \mathrm{OUTQ}_u) \wedge rcvdin_u \vee initiator_u = u$        (S5)
    || $\langle initiator_u, junior_u, rcvdin_u :=$
           $\mathrm{hd}(chnl_{w,v}).init, \mathrm{hd}(chnl_{w,v}).jun$, true if $\mathrm{hd}(chnl_{w,u}).mtype = in$      (S6)
       || $junior_u := min(junior_u, u)^6$ if $rcvdin_u\rangle$                                     (S7)
    || $chnl_{w,u} := \mathrm{tl}(chnl_{w,u})$ if $\mathrm{hd}(chnl_{w,u}).mtype = in \wedge rcvdin_u$                      (S8)
    || $chnl_{u,victim_u} := chnl_{u,victim_u}@abort(u, initiator_u)$ if $detect_u$       $\rangle$             (S9)
▮
$\langle$▮ $u : \mathrm{hd}(chnl_{w,u}).mtype = out ::$
    $\langle || h : h \in \mathrm{HOLD}_u \wedge \neg\, h.flag \wedge h < u \wedge ::$
        $\mathrm{OUTQ}_u := \mathrm{OUTQ}_u \cup \{(initiator_u, junior_u)\}$
                    if $rcvdout_u \wedge \neg\, detect_u \wedge \neg\, exists(initiator_u, \mathrm{OUTQ}_u)$        (S10)
      || $chnl_{u,h}, h.flag := chnl_{u,h}@out(initiator_u, junior_u)$, true
                        if $rcvdout_u \wedge \neg\, detect_u\rangle$                                   (S11)
```

---

[4] $exists(i, Q) \equiv \langle \exists i$: set Q has a message which is initiated by $i\rangle$.

[5] $getjunior(i, Q) \equiv \langle \exists k : k$ is a *junior* of message in Q which is initiated by $i\rangle$

[6] $min(x, y) \equiv min(x, y) := x$ **if** $x \leq y$ ~ y **if** $x > y$.

```
    || $detect_u, victim_u :=$ true, $getjunior(initiator_u, \mathrm{INQ}_u)$
                    if $exists(initiator_u, \mathrm{INQ}_u) \wedge rcvdout_u \vee initiator_u = u$        (S12)
    || $\langle\ initiator_u, junior_u, rcvdout_u :=$
            $\mathrm{hd}(chnl_{w,v}).init, \mathrm{hd}(chnl_{w,v}).jun,$ true if $\mathrm{hd}(chnl_{w,u}).mtype = out$        (S13)
            || $junior_u := min(\ junior_u, u)$ if $rcvdout_u\ \rangle$        (S14)
    || $chnl_{w,u} := \mathrm{tl}(chnl_{w,u})$ if $\mathrm{hd}(chnl_{w,u}).mtype = out \wedge rcvdout_u$        (S15)
    || $chnl_{u,victim_u} := chnl_{u,victim_u}@abort(u, initiator_u)$ if $detect_u\ \ \rangle$        (S16)
▮
$\langle$▮ $u : \mathrm{hd}(chnl_{w,u}).mtype = abort$ ::
    $\langle$ || $r, h : r \in \mathrm{REQ}_u \wedge h \in \mathrm{HOLD}_u$ ::
                $chnl_{u,r}, chnl_{u,h} := chnl_{u,r}@clean\_out(u, initiator_u),$
                $chnl_{u,h}@clean\_in(u, initiator_u)$ if $rca_u\ \rangle$        (S17)
    || $detector_u, initiator_u, rca_u := \mathrm{hd}(chnl_{w,u}).det, \mathrm{hd}(chnl_{w,u}).init,$ true
                    if $\mathrm{hd}(chnl_{w,u}).mtype = abort$        (S18)
    || $chnl_{w,u} := \mathrm{tl}(chnl_{w,u})$ if $(\mathrm{hd}(chnl_{w,u}).mtype = abort \wedge rca_u)$
                    $\vee\ (\mathrm{hd}(chnl_{*,u}).mtype \neq signal \wedge rca_u)$        (S19)
    || $\mathrm{STATE}_u\ : = aborting$ if $\mathrm{hd}(chnl_{initiator_u,u}).mtype = signal \wedge rca_u\ \ \rangle$        (S20)
▮
$\langle$▮ $u : \mathrm{hd}(chnl_{w,u}).mtype = clean\_in$ ::
    $\langle$ || $h : h \in \mathrm{HOLD}_u$ :: $chnl_{u,h} := chnl_{u,h}@clean\_in(victim_u, initiator_u)$
                    if $\neg\ marko_u \wedge rci_u \wedge initiator_u \neq u\ \rangle$        (S21)
    || $victim_u, initiator_u := \mathrm{hd}(chnl_{w,u}).vic, \mathrm{hd}(chnl_{w,u}).init$
                    if $\mathrm{hd}(chnl_{w,u}).mtype = clean\_in \wedge \neg\ rci_u$        (S22)
    || $chnl_{w,u}, \mathrm{INQ}_u := \mathrm{tl}(chnl_{w,u}), \mathrm{INQ}_u \setminus \{(victim_u, *)\}$
                    if $\mathrm{hd}(chnl_{w,u}).mtype = clean\_in \wedge rci_u$        (S23)
    || $chnl_{u,victim_u} := chnl_{u,victim_u}@signal(u)$ if $u = initiator_u \wedge marko_u\ \ \rangle$        (S24)
▮
$\langle$▮ $u : \mathrm{hd}(chnl_{w,u}).mtype = clean\_out$ ::
    $\langle$ || $r : r \in \mathrm{REQ}_u$ :: $chnl_{u,r} := chnl_{u,r}@clean\_out(victim_u, initiator_u)$
                    if $\neg\ marki_u \wedge rco_u \wedge initiator_u \neq u\ \rangle$        (S25)
    || $victim_u, initiator_u := \mathrm{hd}(chnl_{w,u}).vic, \mathrm{hd}(chnl_{w,u}).init$
                    if $\mathrm{hd}(chnl_{w,u}).mtype = clean\_out \wedge \neg\ rco_u$        (S26)
    || $chnl_{u,victim_u}, \mathrm{OUTQ}_u := chnl_{u,victim_u}@signal(u), \mathrm{OUT}_u \setminus \{(victim_u, *)\}$
                    if $\mathrm{hd}(chnl_{w,u}).mtype = clean\_out \wedge rco_u$        (S27)
    || $chnl_{u,victim_u} := chnl_{u,victim_u}@signal(u)$ if $u = initiator_u \wedge marki_u\ \ \rangle$        (S28)
▮
$\langle$▮ $u : v \in \mathrm{REQ}_u \wedge w \in \mathrm{HOLD}_u$ ::
        $marki_u :=$ true if $\mathrm{hd}(chnl_{v,u}).mtype = clean\_in \wedge \mathrm{hd}(chnl_{v,u}).init = u$        (S29)
    || $marki_u :=$ true if $\mathrm{hd}(chnl_{w,u}).mtype = clean\_out \wedge \mathrm{hd}(chnl_{w,u}).init = u\ \ \rangle$        (S30)
end
```

IV. CONCLUDING REMARKS

In this paper we have shown the deadlock detection and resolution algorithm based on the UNITY notion. Deadlock is detected by exactly one process which is a highest priority process in a deadlock cycle. We get rid of retransmiting all detection messages in the resolution phase which improves the performance of the priority based algorithm.

Many deadlock detection and resolution algorithms are incorrect due to the lack of the formalism in their proofs. A major contribution of the paper is that we have given a formal proof of the correctness of the algorithm. The correctness proof is consists of the safety and progress properties of the algorithm, and we have shown that our algorithm holds for these properties. A novel feature of the correctness proof is that it abstracts the program properties and use these properties to prove the correctness of the algorithm.

Other than the program formalism, our algorithm detects a deadlock reduced the time complexity compare to the other edge-chasing algorithms since a deadlock detection message does not have to traverse the entire cycle of *n* message hops when the deadlock is consists of *n* processes. Instead, our algorithm initiates two different type of messages bidirectionally, one to incoming edge and another to outgoing edge of the WFG, and detects a collision by the messages of same initiator. Furthermore, we give stronger restriction to the initiation condition of algorithm to reduce the number of messages generated.

ACKNOWLEDGMENT

The authors would like to thank Ted Herman and Sukumar Ghosh for discussing this work and commenting on earlier draft of this paper.

REFERENCES

[Chandy 82] K. M. Chandy and J. Misra, "A Distributed Algorithm for Detecting Resource Deadlocks in Distributed Systems," *ACM SIGACT-SIGOPS*, Ottawa, Canada, Aug. 1982, pp.157-164.

[Chandy 88] K. M. Chandy and J. Misra, *Parallel Program Design: a Foundation*, Addison-Wesley, Reading, MA, 1988.

[Choud 89] A. N. Choudhary, W. H. Kohler, J. A. Stankovic, and D. Towsley, "A Modified Priority Based Probe Algorithm for Distributed Deadlock Detection and Resolution," *IEEE Transactions on Software Engineering,* Vol.15, No.1, Jan. 1989, pp.10-17.

[Herma 83] T. Herman and K. M. Chandy, "A distributed Procedure to Detect AND/OR Deadlock," The Univ. of Texas at Austin, Tech. Rep., TRH83a-1, 1983.

[Holt 72] R. C. Holt, "Some Deadlock Properties of computer Systems," *ACM Comput. Survey*, Vol.4, No.3, pp.179-196.

[Kshem 91] A. D. Kshemkalyani, M. Singhal, "Invariant-Based Verification of a Distributed Deadlock Detection Algorithm," *IEEE Trans. on Soft. Eng.*, Vol.17, No.8, Aug., 1991, pp.789-799.

[Mitch 84] D. P. Mitchell and M. J. Meritt, "A Distributed algorithm for Deadlock Detection and Resolution," *in Preceedings of the ACM Symposium on Principles of Distributed Computing. ACM,* New York, 1984, pp.282-284.

[Shin 92] Yongtae Shin and Don Epley, "Bidirectional Distributed Deadlock Detection and Resolution Algorithm," The Univ. of Iowa, Tech. Rep., TR92-4, 1992.

[Sinha 85] M. K. Sinha and N. Natarajan, "A Priority Based Distributed Deadlock Detection Algorithm," *IEEE Transactions on Software Engineering,* Vol.SE-11, No.1, January 1985, pp.67-80.

Tight Lower Bounds on Failure Locality of Distributed Synchronization

Manhoi Choy and Ambuj K. Singh
Department of Computer Science
University of California at Santa Barbara
Santa Barbara, California 93106

Abstract

We investigate the fault-tolerance of distributed algorithms in asynchronous message passing systems where the failures of processes are undetectable. We introduce *failure locality* as a measure of the fault-tolerance of distributed algorithms. This measure captures the general idea of allowing part of a system to continue to function despite failures of other parts of the system. We concentrate on two distributed problems, the *dining philosophers problem* and the *committee coordination problem*, and present impossibility results and algorithms with optimal failure locality.

1 Introduction

The ability to tolerate failures is an important design requirement of computer systems in general and of distributed systems in particular as the the detection of failures becomes difficult in such systems. We investigate the effects of faults on distributed synchronization problems that typically require the co-operation of a set of otherwise independent processes in order to ensure certain safety and progress properties. The specific synchronization problems that we study assume an underlying undirected graph that defines the neighborhood of the processes. Inter-process communication is through message-passing and is only possible between neighboring processes. The goal is to design solutions for which the effect of the failure of a process can be confined to its neighborhood. (We assume the *fail-stop* model in which failures are undetectable and failed processes are indistinguishable from processes that are very slow.) In other words, processes should be shielded from the effects of a non-local failure.

We choose the problems of dining philosophers and committee coordination as representatives of the class of distributed synchronization problems. The dining philosophers problem is a generalization of the mutual exclusion problem and can be used to solve many other synchronization problems. The committee coordination problem occurs in languages such as CSP and Ada that are based on synchronous message communication. Though we confine our discussion to these two problems, much of the discussion also extends to other problems involving distributed synchronization.

We use *failure locality* [5] to measure the degree of fault-tolerance. The failure locality of an algorithm denotes the size of the neighborhood that gets affected by a failure. Thus, if an algorithm has a failure locality of m then any process for which there are no failures within a distance of m in the underlying graph executes as if no failures occurred. In other words, any such process continues to meet the specified safety and progress properties.

The main contribution of this paper is the presentation of tight lower bounds on failure locality of solutions to the dining philosophers and the committee coordination problem. This is achieved through the presentation of lower bounds and the design of solutions that achieve the corresponding lower bounds. We view this result as an important step towards a better understanding of distributed algorithms. As a part of the optimal algorithms, we also introduce the idea of a *bounded doorway*. This new kind of a doorway has the interesting property that when coupled with an algorithm ensuring absence of a global starvation, the doorway ensures absence of local starvation *without* adding to the failure locality of the algorithm. This is in contrast to other kind of doorways [5, 12] that add to the failure locality of the underlying algorithm.

The rest of the paper is organized as follows. In Section 2 we present some of the relevant background. In Section 3, we introduce the idea of bounded doorways. In Section 4, we solve the generalized dining philosophers problem with an optimal failure locality. In Section 5, we consider the committee coordination problem. We present a solution with an optimal failure locality for the binary version of the problem and we discuss an almost optimal solution for the general version. We conclude with a brief discussion in Section 6.

2 Relevant Background

The notion of locality depends on the kind of problem being considered. Locality has been studied by many authors [1, 13, 17, 18] in the area of distributed systems. Linial [13] considered the problem of computing functions that do not require global communication. To compute a function, each process starts with some local data. During the computation, information is exchanged between neighboring processes. Finally, each of the processes terminates and returns a value. A function is said to be computed *locally* [1] if it can be computed in a time less than the diameter of the network. Some of these functions include finding a *maximal independent set* [1, 10, 13, 14] or node (and edge) colorings of a network [1, 9]. The locality we are interested in is defined for a different set of problems. We investigate the locality of distributed synchronization problems that do not require processes to terminate and return results. Two representative problems that have drawn a lot of interest in this area are the *dining philosophers* problem[7, 15] and the *committee coordination* problem [4].

Many solutions to the dining philosophers problem appear in the literature [2, 4, 5, 15, 18]. Styer and Peterson [18] are perhaps the first to consider the locality of this problem. They measure the locality of an algorithm by the length of the longest waiting chain of processes that may be formed in the system. We formalize this idea of waiting chains to the notion of *failure locality.* For the dining philosophers problem, failure locality is defined as the smallest number m such that a philosopher is free from starvation as long as no process within a distance m from it in the communication network fail. The algorithms obtained by Styer and Peterson have failure locality of 3 and $log\delta$ where δ is maximum degree of the underlying communication graph. Recently, we considered the problem of achieving a good failure locality in conjunction with the problem of achieving a good response time and a good message complexity [5]. We presented solutions that achieved a failure locality of 4 while maintaining a quadratic response time and a quadratic message complexity. The paper also contains an algorithm that achieves a failure locality of 3 but with exponential response time and message complexity. However, the question of what is the best achievable failure locality still remained open.

The committee coordination problem has been solved by many researchers [3, 4, 5, 11, 16, 17]. Sistla [17] calls a solution to the problem to be *real time* if its complexity is independent of the size of the communication network. This means that the executions of the processes should have local properties and should be independent of the entire network. The author also investigates the formation of waiting chains and then uses this to derive a lower bound on the response time. Recently, we presented solutions to the committee coordination that obtained complexity measures similar to that for the dining philosophers problem [5]. But, once more the problem of the optimal failure locality remained open.

3 Bounded Doorways

Solutions to problems requiring the absence of local starvation are often composed of two parts [5, 12, 18]. The first part satisfies all the requirements of the problem except that it ensures the absence of global starvation (i.e., no deadlocks). The second part of the solution then has the responsibility of meeting the stronger requirement of absence of local starvation (i.e., progress with respect to each process). Schemes of this kind that translate absence of global starvation of an underlying solution to absence of local starvation for the composite solution are often referred to as *doorways*. In other words, a doorway D takes a program F ensuring absence of global starvation, and produces a composite program consisting of F and D such that local starvation does not occur.

Various kinds of doorways have been defined and used in the literature. A basic kind called the *first-come-first-served* doorway was defined and used by Lamport [12] in the context of the mutual exclusion problem. This doorway is required to be wait-free i.e., it is required to consist of a finite sequence of non-waiting statements. Though useful for shared variable systems, this doorway cannot be used for message based systems as the doorway entering part would have to consist of waiting to receive messages in order to enforce any kind of a first-come-first-served behavior. A variant of this kind of doorways was used by Styer and Peterson to solve the dining philosophers problem in a message based model similar to ours [18]. In their scheme neighboring processes executed a P operation on a common semaphore (implemented by passing messages) while entering the doorway and a V operation while exiting the doorway. Two other kinds of doorways called *single doorway* and *double doorway* were introduced by us in [5] in order to avoid unbounded overtaking. All the above doorways require processes outside the doorway to wait for processes inside the doorway and this increases the failure locality of the composite solution by at least one.

We introduce a new kind of a doorway that does not increase the failure locality of the underlying computation. This doorway, referred to as a *bounded doorway*, is implemented by dynamically assigning a set of distinct IDs to the processes. The IDs must be distinct so that they can be used for resolving conflicts that occur during the execution of the algorithm. We assume that the underlying solution resolves conflicts in favor of the process with a smaller ID. In the doorway algorithm, a process that has just made progress selects (in its doorway exiting section) one of its neighbors to perform an ID exchange. This exchange results in the neighbor having a smaller ID (i.e. higher priority) then before. An ID exchange involves sending an *exchg* message to the selected neighbor, waiting for an *exchg-ack* message to be returned, and broadcasting of the new ID to other processes. The process receiving an *exchg* message agrees to the exchange of IDs only if this will lead to a smaller ID and the process is not currently participating in another exchange. In order to ensure that local starvation does not occur, processes maintain a circular queue of neighbors and choose the candidate process for the ID exchange from the head of the queue. Thus, a process that makes progress infinitely often will eventually have a higher ID (and consequently a lower priority) than all its neighbors. This will allow the neighbors to make progress. The ID exchange mechanism also maintains the uniqueness of the process IDs and does not interfere with the underlying computation as the underlying computation is restarted each time the exchange of IDs affects the existing conflict resolution.

4 The Dining Philosophers Problem

4.1 Problem Description

The problem consists of a set of processes located at the nodes of an underlying conflict graph. Each process may be in one of three states: *thinking*, *hungry*, or *eating*. Initially every process is thinking. A thinking process becomes hungry when it needs to access the resources on the adjoining edges. A process stays hungry until it is allowed to eat by the dining philosophers algorithm. A process accesses the resources during its eating period. After that an eating process transits to the thinking state. A dining philosophers algorithm should satisfy two requirements: *mutual exclusion* which requires that neighboring processes do not eat at the same time and *no starvation* which requires that every hungry process eventually gets a chance to eat.

4.2 Impossibility Result

Theorem 1 *There does not exist a solution to the dining philosophers problem with a failure locality of 1.*

Proof: Suppose there exists a solution with a failure locality of one. Consider the problem instance of 3 processes i, j, and k where i and k are neighbors of j but i and k are not neighbors. Consider the following execution in which none of the processes fail.

Initially all the processes are thinking. Process i becomes hungry first. Since thinking processes may remain thinking forever, there exists an execution in which i starts eating while j and k are still thinking. Now, assume that while i is eating both j and k become hungry. It is possible that process i fails undetectably while eating and therefore in order to ensure a failure locality of one, process k should be allowed to eat. Thus, there exists a future state in which i and k are eating and j is hungry. Now, suppose that i finishes eating and becomes hungry again. By the same argument as before, there exists an execution in which k remains eating, j remains hungry, and i starts eating. Now, let process k stop eating and become hungry again. This leads us back to a state that had existed earlier in the history. Note that process j remains hungry in all the intermediate states. This scenario may repeat infinitely often, giving us the desired contradiction as process j remains hungry forever while none of its neighbors have failed. □

4.3 An Optimal Algorithm

The algorithm is a composition of two components: a component consists of a bounded doorway and another component that ensures mutual exclusion among neighboring processes and absence of global starvation. Mutual exclusion is ensured by associating forks with edges and maintaining the invariant that an eating process has all the neighboring forks. In order to ensure the absence of global starvation, conflicting requests to the same fork are resolved in favor of the smaller ID. Since this may lead to the formation of long waiting chains (which in turn results in poor failure locality), processes break such long chains by releasing forks if they suspect that long waiting chains are being formed.

```
Initially:
    $ID_i = i$; $\langle \forall j \in N_i : \neg requested_{ij} \rangle$;
    $\langle \forall j \in N_i : fork_{ij} = (ID_i < ID_j) \wedge ID_{ij} = ID_j \rangle$;
    insert all elements of $N_i$ to $Q_i$ in any order;
    $state_i = thinking$;
    $pending_i = \{\}$;
    $ack_i = true$;
Define:
    $all\text{-}forks_i \equiv \langle \forall j \in N_i : fork_{ij} \rangle$;
    $all\text{-}low\text{-}forks_i \equiv \langle \forall j \in N_i : ID_{ij} < ID_i \Rightarrow fork_{ij} \rangle$;
    $eating\text{-}or\text{-}ready_i \equiv$
        $(state_i = eating \vee (state_i = hungry \wedge all\text{-}forks_i \wedge ack_i))$;
    $release\text{-}high_i \equiv (state_i = thinking \vee \neg ack_i \vee \neg all\text{-}low\text{-}forks_i)$;

/* The main procedure */
repeat:
    ⟨non-critical section⟩;
    $state_i := hungry$;
    wait until $ack_i$;
    request-forks;
    wait until $all\text{-}forks_i$;
    $state_i := eating$;
    ⟨critical section⟩;
    $state_i := thinking$;
    release-id;
    release-forks;
forever;
```

Figure 1: Optimal Algorithm for Dining Philosophers

However, the process with the smallest ID does not release any forks. This ensures the absence of global starvation. The ID exchange mechanism due to the bounded doorway ensures that a continuously hungry process will eventually have a smaller ID than its competing neighbors and will be able to eat subsequently. The composite algorithm thus also ensures freedom from local starvation.

The algorithm is shown in Figure 1. When a process becomes hungry, it attempts to collect missing forks by sending request messages, as in procedure *request-forks*, to the corresponding neighbors. A process receiving a request decides whether it is going to release the fork to the requester or not depending on its state and the relative order of the IDs. If the fork is not released then the ID of the requester is saved in a set $pending_i$ for future consideration.

The procedure *send-fork* is used to send forks to neighbors. Boolean flags are sent along with forks to reflect the status of the sender. The flag is true iff the sender has been preempted and still needs the fork. A received fork may induce one of two actions. First, if the reception of the fork completes the collection of all the low forks then request for missing high forks are made, if necessary, by calling procedure *request-high-fork*. Second, if the fork just received is a high fork, it has a true flag, and not all the low forks are collected then this high fork is returned to the sender immediately.

From the way forks are sent and released, it is possible to show that if a process v is holding a fork from its requesting neighbor u then either 1) process v has failed, or 2) process v is going to eat immediately, or 3) process v is a low neighbor of u and v has collected all its low forks. From these, one can show that the length of any waiting chain in the system is at most two.

The implementation of the bounded doorways is based on an exchange of IDs as discussed earlier. When a process P_i is requesting for an ID exchange, the variable ack_i is set to false and P_i has to release forks on request. The variable Q_i in Figure 1 is a circular queue storing the set of neighboring processes N_i of process P_i with whom P_i exchanges ID. The remaining description of its implementation and the proof of correctness of the algorithm is postponed to the full paper.

```
On receiving message $\langle request \rangle$
    from $P_j$ such that $ID_{ij} > ID_i$:
    $pending_i := pending_i \cup \{j\}$;
    if $(fork_{ij} \wedge \neg eating\text{-}or\text{-}ready_i \wedge release\text{-}high_i)$ then
        send-fork(j) fi
```

```
On receiving message $\langle request \rangle$
    from $P_j$ such that $ID_{ij} < ID_i$:
    $pending_i := pending_i \cup \{j\}$;
    if $(fork_{ij} \wedge \neg eating\text{-}or\text{-}ready_i)$ then
        send-fork(j);
        release-high-forks fi
```

```
On receiving message $\langle fork, flag \rangle$ from $P_j$:
    $fork_{ij} := true$;
    $requested_{ij} := false$;
    if $flag$ then $pending_i := pending_i \cup \{j\}$;
    if $(all\text{-}low\text{-}forks_i \wedge ack_i \wedge state_i = hungry)$ then
        request-high-forks;
    elsif $(flag \wedge ID_{ij} > ID_i \wedge \neg all\text{-}low\text{-}forks_i)$ then
        send-fork(j) fi
```

```
On receiving message $\langle exchg, ID \rangle$ from $P_j$:
    if $(ack_i \wedge ID_i > ID)$ then
      send ⟨exchg-ack, $ID_i$⟩ to $P_j$;
      $\langle \forall k \in N_i$ : send ⟨new-ID, ID⟩ to $P_k \rangle$;
      $ID_{ij} := ID_i$;
      $ID_i := ID$;
      if $(all\text{-}low\text{-}forks_i \wedge ack_i \wedge state_i = hungry)$ then
          request-high-forks;
   else send ⟨exchg-ack, ID⟩ to $P_j$ fi
```

```
On receiving message $\langle exchg\text{-}ack, ID \rangle$ from $P_j$:
    if $(ID \neq ID_i)$ then
        $\langle \forall k \in N_i$ : send ⟨new-ID, ID⟩ to $P_k \rangle$; fi
    $ID_i := ID$;
    $ack_i := true$
```

```
On receiving message $\langle new\text{-}ID, ID \rangle$ from $P_j$:
    $ID_{ij} := ID$;
    if $(ID_{ij} < ID_i \wedge j \in pending_i \wedge fork_{ij}$
        $\wedge \neg eating\text{-}or\text{-}ready_i)$ then
        send-fork(j);
        release-high-forks;
   elsif $(all\text{-}low\text{-}forks_i \wedge ack_i \wedge state_i = hungry)$ then
        request-high-forks fi
```

```
Procedure request-high-forks
    for each $(j \in N_i)$ do
        if $(ID_{ij} > ID_i \wedge \neg requested_{ij}$
            $\wedge \neg fork_{ij})$ then
            send ⟨request⟩ to $P_j$;
            $requested_{ij} := true$ fi;
    od
end request-high-forks
```

```
Procedure request-forks
    for each $j \in N_i$ do
        if $\neg fork_{ij} \wedge \neg requested_{ij}$ then
            send ⟨request⟩ to $P_j$;
            $requested_{ij} := true$ fi;
    od
end request-forks
```

```
Procedure release-forks
    for each $j \in pending_i$ do
        send-fork(j) od
end release-forks
```

```
Procedure release-high-forks
    for each $k \in pending_i$ do
        if $(fork_{ik} \wedge ID_{ik} > ID_i)$ then
            send-fork(k) fi;
    od
end release-high-forks
```

```
Procedure release-id
    tmp := next;
    if $tmp \neq null$ then
        send ⟨exchg, $ID_i$⟩ to $P_{tmp}$;
        $ack_i := false$ fi
end release-id
```

```
Procedure send-fork(j)
    $fork_{ij} := false$;
    $pending_i := pending_i - \{j\}$;
    if $(state_i = hungry \wedge \neg requested_{ij}$
        $\wedge ack_i)$ then
        send $\langle fork, true \rangle$ to $P_j$;
        $requested_{ij} := true$;
    else
        send $\langle fork, false \rangle$ to $P_j$ fi
end send-fork
```

Figure 1 (contd.): Message Handlers.

5 The Committee Coordination Problem

5.1 Problem Description

The problem consists of a set of processes and a set of interactions. Each process has a predetermined set of interactions that it may participate in and each interaction has a predetermined set of processes that co-operate to execute it. Processes that participate in the same interaction are physically close to each other and can communicate directly. A process or an interaction is in one of three states: *idle*, *ready*, and *commit*. Initially all processes are *idle*. From time to time, a process becomes *ready* to take part in any of the interactions associated with it and it remains *ready* until one of the interactions is committed. An interaction is *ready* to be executed if all the processes associated with it are *ready*, an interaction is in the *commit* state if all the processes participating in it are committed to perform it, and an interaction is *idle* otherwise. During the execution of an interaction, the interaction and all the processes associated with the interaction are in the state *commit*. The commitment of each interaction eventually terminates, at which point the interaction and the processes become *idle* again.

We say that two interactions *conflict* with each other if they share a common process. Based on this conflict relation between interactions, we define a *conflict graph* as follows: we represent each interaction by a node and place an edge between two nodes if the corresponding interactions conflict. As mentioned earlier, the state transitions of a process occur in the order *idle* to *ready* to *commit* and back to *idle*. The state transitions of an interaction are similar except for the fact that an interaction may transit to state *idle* directly from the state *ready*. This is because two neighboring interactions may be ready at the same time and committing one changes the state of the other to *idle*. A solution to the committee coordination problem should satisfy two safety requirements, *synchronization* and *mutual exclusion*, and one progress requirement, *weak interaction fairness*. Synchronization requires that the commitment of an interaction is only started if it is ready. Mutual exclusion requires that no two interactions sharing a common process commit simultaneously. Weak interaction fairness requires that if an interaction is continuously ready then it will eventually be committed.

We define the failure locality of an algorithm for the committee coordination problem as the minimum number m such that for any interaction, as long as interactions within a distance m (measured in the conflict graph) do not fail, the three requirements of the problem are still satisfied.

5.2 Impossibility Result

Theorem 2 *There does not exist a solution to the committee coordination problem that achieves a failure locality of 0.*

Proof: The proof is by reduction to the well-known result about the impossibility of distributed consensus with one faulty process [8]. Assume that there exists an algorithm that achieves a failure locality of 0. In other words, the algorithm supposedly ensures that as long as processes in an interaction are ready and do not fail, the interaction or one of its neighbors commit eventually. We use such an algorithm to solve the distributed consensus problem for three processes i, j, and k.

In the consensus problem every process has an initial value and all non-failed processes have to agree on a common value that was one of the initial values. For the case of three processes that we consider here, we form three committees $\mathcal{I}=\{i,j\}$, $\mathcal{J}=\{j,k\}$, and $\mathcal{K}=\{k,i\}$ and devise a mechanism by which the commitment of an interaction (e.g. $\mathcal{I}$) implies that a consensus is reached on the initial value of the corresponding process (in this case i).

Every process first broadcasts its initial value to the other processes and then becomes ready to participate in any interaction of which it is a member. The underlying committee coordination algorithm commits exactly one of the interactions $\mathcal{I}$, $\mathcal{J}$, or $\mathcal{K}$. The processes that are participating in the committed interaction broadcast its identity to the third process. Finally, the processes agree on the initial value of process corresponding to the committed interaction.

Since processes broadcast their initial values before becoming ready, the value corresponding to a committed interaction is available at all the processes. Since at most

```
Initially:
    $ID_i = i$;
    $\langle \forall j \in N_i : ID_{ij} = ID_j \rangle$;
    insert all elements of $N_i$ to $Q_i$ and $L_i$;
    $state_i = idle$;
    $ack_i = true$;
```

```
/* The main procedure */
repeat:
    $\langle idle\ region \rangle$;
    $\langle \forall j \in N_i : requested_{ij} := false \rangle$;
    wait until $ack_i$;
    wakeup-neighbors;
    request-commitment;
    wait until $state_i = commit$;
    $\langle perform\ interaction \rangle$;
    $state_i := idle$; release-id;
forever;
```

Figure 2: Optimal Algorithm for Committee Coordination

one process fails, at least one interaction will be ready to be committed. By the assumed failure locality of the underlying committee coordination algorithm, at least one interaction commits. Therefore, at least one process is able to broadcast the identity of the committed interaction. As a result, all the non-failed processes agree on a common value that was one of the initial values. □

5.3 The Binary Case

We present an optimal solution to the binary version of the committee coordination problem, i.e., each interaction is between exactly two processes. When a process becomes ready, it attempts to commit interactions by sending request messages to neighbors. In order to avoid a cycle in which each process is requesting the next process, we only allow processes with smaller IDs to make requests. When a process receives a request, it replies with a yes message if it is ready and has not sent a yes message to another requester. Otherwise, it saves the requester in the set *pending*. If a process P_i receives a yes message from P_j then it commits the corresponding interaction provided it is still ready and has not responded with a yes message to another process. Subsequently, if P_i does decide to commit the interaction, then it sends a commit message P_j and transits to state commit. On receiving this commit message P_j also transits to state commit. Otherwise, P_i sends a release message to P_j. On receiving this release message, P_j attempts to commit an interaction with one of the pending processes in set $pending_j$ by sending a *wakeup* message.

Based on the above scheme of sending messages, it is possible to show that the requirements of synchronization and mutual exclusion are met, that waiting chains of length more than 2 (and consequently, of length more than 1 between interactions) do not form, and that global starvation does not occur. To ensure weak interaction fairness (or absence of local starvation), a bounded doorway similar to that in the algorithm to the dining philosophers problem is included. However, IDs are exchanged only after an interaction is committed and the set of processes N_i that may be chosen for ID exchange by process P_i is the set of processes that P_i may commit with. The complete algorithm is shown in Figure 2 and its proof of correctness is postponed to the full paper.

5.4 The General Case

The algorithm for the binary case does not extend to the general case directly to give an optimal algorithm. The main reason is that even if one of the processes in an interaction has received the smallest ID in the entire system, there is still no guarantee that the

```
Procedure request-commitment
    if $state_i = ready \wedge \neg bound_i \wedge ack_i$ then
        $\langle \forall j \in N_i : ID_{ij} > ID_i \wedge \neg requested_{ij}$:
            send $\langle request \rangle$ to $P_j$;
            $requested_{ij} := true$; $\rangle$
    fi
end request-commitment

On receiving message $\langle request \rangle$ from $P_j$:
    if ($state_i = ready \wedge \neg bound_i$) then
        send $\langle yes \rangle$ to $P_j$;
        $bound_i := true$;
    else
        $pending_i := pending_i \cup \{j\}$; fi

On receiving message $\langle yes \rangle$ from $P_j$
    if ($state_i = ready \wedge \neg bound_i$) then
        send $\langle commit \rangle$ to $P_j$;
        $state_i := commit$;
    else
        send $\langle release \rangle$ to $P_j$;
        $requested_{ij} := false$; fi

On receiving message $\langle release \rangle$ from $P_j$
    $bound_i := false$;
    request-commitment;
    wakeup-neighbors;

On receiving message $\langle wakeup \rangle$ from $P_j$
    if ($state_i = ready$) then
        $requested_{ij} := false$;
        request-commitment; fi
```

```
On receiving message $\langle commit \rangle$ from $P_j$:
    $state_i := commit$;

On receiving message $\langle new\text{-}ID, ID \rangle$ from $P_j$:
    $ID_{ij} := ID$;
    request-commitment

On receiving message $\langle exchg, ID \rangle$ from $P_j$:
    if ($ack_i \wedge ID_i > ID$) then
        send $\langle$exchg-ack, $ID_i \rangle$ to $P_j$;
        $\langle \forall k \in N_i$ : send $\langle$new-ID, ID$\rangle$ to $P_k \rangle$;
        $ID_{ij} := ID_i$;
        $ID_i := ID$;
        request-commitment;
    else send $\langle$exchg-ack, ID$\rangle$ to $P_j$; fi

On receiving message $\langle exchg\text{-}ack, ID \rangle$ from $P_j$:
    if ($ID \neq ID_i$) then
        $\langle \forall k \in N_i$ : send $\langle$new-ID, ID$\rangle$ to $P_k \rangle$; fi
    $ID_i := ID$;
    $ack_i := true$;

Procedure wakeup-neighbors
    for each ($j \in pending_i$) send $\langle wakeup \rangle$ to $P_j$;
    $pending_i = \{\}$;
end wakeup-neighbors

Procedure release-id
    tmp := next;
    if $tmp \neq null$ then
        send $\langle$exchg, $ID_i \rangle$ to $P_{tmp}$;
        $ack_i := false$; fi
end release-id
```

Figure 2 (contd.): Message Handlers.

interaction will eventually be committed. Nevertheless, it is possible to obtain an almost optimal algorithm (with a failure locality of 2) for the general case. The basic idea is to solve the committee coordination problem with an underlying algorithm for the dining philosophers problem as in [4, 5]. For this we use the dining philosopher algorithm presented in Section 4. It can be shown that the failure locality of the resulting solution is 2. The problem of whether a failure locality of 1 is achievable for the general version of the problem remains open.

6 Conclusion and Discussion

Both the optimal fault-tolerant algorithms presented in this paper are based on the idea of bounded doorways. Bounded doorways are interesting because they can ensure individual progress from a given algorithm that achieves global progress without increasing the failure locality of the algorithm. Recently, we have used the idea of bounded doorways to construct a solution to the mutual exclusion algorithm that adapts to the load in the system [6].

The response time of the solutions presented here appears to be exponential in δ, the degree of the underlying conflict graph. We are currently pursing the construction of solutions with a polynomial response time that also achieve an optimal failure locality.

References

[1] B. Awerbuch, A. B. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30th IEEE Annual Symposium on Foundations of Computer Science*, pages 364–369, 1989.

[2] B. Awerbuch and M. Saks. A dining philosophers algorithm with polynomial response time. In *Proceedings of the 31st Symposium on Foundations of Computer Science*, pages 65–74, 1990.

[3] G. N. Buckley and A. Silberschatz. An effective implementation for the generalized input-output construct of csp. *ACM Trans. Program. Lang. Syst.*, 5(2):223–235, (April 1982).

[4] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation.* Addison Wesley Pub. Co., Reading, Mass., 1988.

[5] M. Choy and A. K. Singh. Efficient fault tolerant algorithms for resource allocation in distributed systems. 1992, submitted to Journal. A preliminary version appears in *the Proceedings of the 21th Annual ACM Symposium on Theory of Computing*, 1992, pages 593-602.

[6] M. Choy and A. K. Singh. Adaptive algorithms for mutual exclusion. in preparation.

[7] E. W. Dijkstra. Hierarchical ordering of sequential processes. *Acta Inf.*, 1:115–138, 1971.

[8] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[9] A.V. Goldberg and S.A. Plotkin. Parallel $(\delta + 1)$ coloring of constant-degree graphs. *Information Processing Let.*, 25:241–245, 1987.

[10] R. M. Karp and A. Wigderson. A fast parallel algorithm for the maximal independent set problem. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 331–335, 1987.

[11] D. Kumar. An algorithm for n-party synchronization using tokens. In *Proceedings of the 10th International Conference on Distributed Computing Systems*, pages 320–327, 1990.

[12] L. Lamport. The mutual exclusion problem: Part i and ii. *J. ACM*, 33(2):313–348, (April 1986).

[13] N. Linial. Distributive algorithms – global solutions from local data. In *Proceedings 28th IEEE Annual Symposium on Foundations of Computer Science*, pages 331–335, 1987.

[14] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15:1036–1052, 1986.

[15] N. Lynch. Fast allocation of nearby resources in a distributed system. In *Conference Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 70–81, 1980.

[16] S. Ramesh. A new and efficient implementation of multiprocess synchronization. In *PARLE Parallel Architectures and Languages Europe. Lecture Notes in Computer Sciences 259*, pages 387–401, Springer-Verlag, Berlin, 1987.

[17] A. P. Sistla. Distributed algorithms for ensuring fair interprocess communications. In *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, pages 266–277, (Vancouver, Aug. 1984).

[18] E. Styer and G. Peterson. Improved algorithms for distributed resource allocation. In *Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 105–116, (Toronto, Aug. 1988).

Testing and Debugging Distributed Programs Using Global Predicates

S. Venkatesan*
Computer Science Program
The University of Texas at Dallas
Richardson, TX 75083-0688
venky@utdallas.edu

Brahma Dathan
Department of Computer Science
University of Wyoming
Box 3682
Laramie, WY 82071

Abstract

Testing and debugging programs are more involved in distributed systems than in uni-processor systems because of the presence of the communication medium and the inherent concurrency. This paper presents a practical approach to testing and debugging distributed programs based on global predicates. Several issues related to global predicates are addressed and efficient algorithms are presented. The techniques can be implemented easily in distributed systems.

1 Introduction

Although distributed systems have several advantages over centralized systems, they are harder to design and implement [3]. Much work has been done in the area of developing protocols for solving many problems distributively, but the problem of validating or testing an implementation of these protocols remains unsolved. Recent work on protocol validation [4, 9] focuses on developing protocols that avoid communication problems such as deadlocks, unspecified reception, non-executable transition, etc. However, several other errors may exist in an implementation of a protocol. This paper presents a practical testing and debugging methodology using global predicates. This methodology can be easily implemented in distributed (message-passing) systems.

1.1 Some Basic Concepts, Notations, and Definitions

The distributed system consists of n processes, uniquely numbered from 1 to n. Each process resides on a unique processor and there is a one-to-one mapping between processors and processes. The processes are denoted by $P_1 \dots P_n$. Processes communicate with each other by passing messages only. The system is *asynchronous*: the time taken by processors to complete an instruction and the message delivery times are arbitrary but finite.

---

*This research was supported in part by NSF under Grant No. CCR-9110177 and by the Texas Advanced Technology Program under Grant No. 9741-036.

The Notion of Events All processes in the system are "event-driven" – an event occurs when a process receives a message. In any distributed system with unknown processor speeds and communication delays, processes must be event-driven. An event may cause the process to change its state (modify some of its variables) and/or send zero or more messages to other processes. Let e^i_j denote an event numbered j in P_i. We number events within each process starting with 0.

Message Logs An omniscient observer of a distributed program in execution can record when messages are sent, when they are received, and what effects they have. Such information can be represented by a directed acyclic graph called a *log*, with exactly one node for each event. (Label each node with the id of the event it represents.) The edges of the graph are such that there is a path from e^i_j to e^i_k if $j < k$. We also draw an arc from e^i_j to e^k_l if the event e^i_j results in the transmission of a message that is received by P_k causing the event e^k_l. The graph is easier to analyze if all nodes corresponding to events in a single process are aligned horizontally, generating a "process line" for each process. Due to the way we have aligned vertices and drawn edges in this graph, the horizontal axis may be thought of as representing time. In our figures, the time increases as we move to the right on the horizontal axis.

Formal Definitions Let e and e' be any two events in the distributed system such that either they both occur in the same process and e occurs before e' or e corresponds to the sending of a message m that is received during the event e'. We then say that e is a predecessor of e' and e' is a successor of e and denote this by writing $e \preceq e'$. Note that an event is a predecessor of itself. The fact that e is not a predecessor of e' is denoted by writing $e \not\preceq e'$. If $e \not\preceq e'$ and $e' \not\preceq e$, the two events are said to be *independent.* If $e \preceq e'$ and $e \neq e'$, we then say that e is a proper predecessor of e' (and e' is a proper successor of e) and write $e \prec e'$. To denote that e is not a proper predecessor of e', we write $e \not\prec e'$.

A *spectrum* is a sequence of consecutive events in a process. At its extreme, the spectrum may be just a single event or it may include all of the events in the process.

Let S_i be a spectrum in P_i. An event e^i_j in S_i is the last event in S_i if no event in S_i is a proper successor of e^i_j; e^i_j is said to be first event in S_i if no event in S_i is a predecessor of e^i_j. The last (or first) event of S_i is denoted by $last(S_i)$ (or $first(S_i)$). Since a spectrum may include all the events of a process, last and first are well defined for processes. If e^i_j is not the last event in P_i, $next(e^i_j)$ is event e^i_{j+1}.

Let e^i_j be an event of process P_i. The *first concurrent event* of e^i_j at process P_k, denoted by $\mathcal{FCE}_k(e^i_j)$ is that event e^k_l such that $e^k_l \prec e^i_j$ and for any e^k_m such that $e^k_l \prec e^k_m$, $e^k_m \not\prec e^i_j$. That is, $\mathcal{FCE}_k(e^i_j)$ is the earliest event of P_k such that none of the successors of $\mathcal{FCE}_k(e^i_j)$ is a predecessor of e^i_j. If no event of P_k precedes e^i_j ($e^k_n \not\prec e^i_j$ for all $n \geq 0$), then $\mathcal{FCE}_k(e^i_j) = e^k_0$.

The *last concurrent event* of e^i_j (of process P_i) at process P_k, denoted by $\mathcal{LCE}_k(e^i_j)$ is the latest event e^k_l such that if $next(e^i_j)$ exists, then $next(e^i_j) \not\prec e^k_l$. If $next(e^i_j)$ is not defined, $\mathcal{LCE}_k(e^i_j)$ is the last event in P_k.

The *concurrency set* of an event e^i_j at process P_k is the spectrum S_k with $first(S_k) = \mathcal{FCE}_k(e^i_j)$ and $last(S_k) = \mathcal{LCE}_k(e^i_j)$.

A *global state*, or simply *state* [1], of the system is a tuple with exactly one event for each process such that each component of the tuple is in the concurrency set of every other component in the tuple.

Representation of the Log Each process stores information about events taking place within the process. When a message is received, store the id of the sender thus marking the beginning of the event[1]. When a message is sent, store the id of process to whom this is being sent. After all the messages sent for this event are recorded, mark the end of the event.

Representing the distributed execution in the above manner is attractive because its storage requirements are not high. Even for a test where each process receives 1 million messages, the requirement is only around 2 MBytes (if less than 65536 processes are involved in the test), a relatively small amount even for a personal computer!

1.2 Issues for Investigation

A single execution of a distributed algorithm can be captured by a message log, but a single log represents several potential global executions and each one must be examined for errors. Many of the issues in testing and debugging can be translated into corresponding problems on a log. To test if the execution is correct, we address the following questions such as (a) Is there a potential incorrect execution? (b) Given an incorrect execution, how can we bring the global computation to a particular global state that we wish to examine? (b) To aid debugging, we wish to answer questions such as "What is the earliest state where a given predicate is true ?".

The number of possible global states may be exponential. Thus, examining all possible states is impractical in large systems.

2 Evaluation of Predicates

To determine whether a distributed algorithm executed correctly, one could evaluate several predicates. The following appear to be useful.
POSSIBLY(Φ): Does there exist a global state where Φ holds ? If Φ represents an error condition, then POSSIBLY(Φ) represents a possible occurrence of an error. For example, if the distributed program implements a solution to the mutual exclusion problem, then $\Phi = (P_1 \text{ is in critical section}) \wedge (P_2 \text{ is in critical section})$ denotes an error.
DEFINITELY(Φ): Is it the case that no matter how the events may have taken place relatively, Φ holds in one of the possible global states of the computation ?

---

[1] Several changes to local variables may take place. But, the message log does not contain that information.

FIRST(Φ): If POSSIBLY(Φ) is true, then find the earliest global state where Φ holds. If Φ is an error condition, then for debugging, the user may want to bring the computation to the first global state where Φ holds.

LAST(Φ): This is same as FIRST(Φ) except that we are interested in finding the last global state where Φ holds.

Any boolean expression can be expressed in a standard form known as conjunctive normal form (CNF). A boolean expression is in conjunctive normal form if it is a conjunction of expressions $C_1, C_2, \ldots, C_m$, where each conjunct C_i is itself a disjunction of literals (either a variable or the negation of a variable). In this section, we show how to evaluate the properties DEFINITELY, POSSIBLY, LAST, and FIRST for predicates in conjunctive normal form where each C_i may be any boolean expression involving constants and variables that are local to process P_j.

Expressions in CNF can be used to test a variety of conditions. For example, for testing a distributed program that implements mutual exclusion, we could test the condition that only one process is in the critical section at any given time by testing for the property $\neg$ POSSIBLY $((mutex_1 = true) \wedge (mutex_2 = true))$.

2.1 Testing POSSIBLY(Φ)

The technique is based on the notion of *projections.* Let S_i be a spectrum at site P_i. The *projection* of S_i on process P_j, denoted by $\pi(S_i \rightarrow P_j)$ is a spectrum S_j at P_j such that $first(S_j) = \mathcal{FCE}_j(first(S_i))$ and $last(S_j) = \mathcal{LCE}_j(last(S_i))$. Intuitively, $\pi(S_i \rightarrow P_j)$ is the spectrum of events of P_j that are in concurrency set of at least one event of S_i. Note that $\pi(S_i \rightarrow P_i) = S_i$.

Let S_1^i and S_2^i be two spectra at P_i. The intersection of S_1^i and S_2^i is the spectrum S_3^i at P_i, possibly empty, such that e is included in S_3^i iff e is included in both S_1^i and S_2^i. The intersection operation is denoted by $\cap$. The appendix contains a centralized algorithm for finding the intersection of projections, and the running time of the algorithm is $O(n^3)$. See [8] for a distributed implementation that uses $O(n^3)$ messages.

2.1.1 A Centralized Algorithm

We now describe a technique for determining POSSIBLY(Φ) assuming that the complete log is available. (In the distributed case, each site has only part of the log [8].)

Lemma 1 *Let* $S_1, \ldots, S_n$ *be spectra on processes* $P_1, \ldots, P_n$ *respectively. Let* $I_1, I_2, \ldots, I_n$ *be spectra such that* $I_i = \bigcap_{j=1}^{n} \pi(S_j \rightarrow P_i)$, $1 \leq i \leq n$. *Then,* $(first(I_1), \ldots, first(I_n))$ *and* $(last(I_1), \ldots, last(I_n))$ *are global states.*

Proof. We prove by contradiction that $(last(I_1), \ldots, last(I_n))$ is a global state. Assume to the contrary. Then there exist processes P_i and P_j such that $last(I_i)$ and $last(I_j)$ are not in each other's concurrency sets. There are four cases to consider.

1. Evaluate C_i at each event of process P_i and let S_i be single resulting spectrum at P_i.
2. If Φ does not contain a conjunct for some process P_j, take $first(S_j)$ = first event of P_j and $last(S_j)$ = last event of P_j.
3. Each spectrum S_i is projected on to all of the processes.
4. At each site i, evaluate $I_i = \bigcap_{j=1}^{n} \pi(S_j \rightarrow P_i)$.
5. If I_i is empty for at least at one process P_i, then POSSIBLY(Φ) is false. Otherwise, it is true.

Figure 1: Centralized algorithm for determining POSSIBLY(Φ)

1. $last(I_i) \prec \mathcal{FCE}_i(last(I_j))$.
2. $\mathcal{LCE}_j(last(I_i)) \prec last(I_j)$.
3. $last(I_j) \prec \mathcal{FCE}_j(last(I_i))$.
4. $\mathcal{LCE}_i(last(I_j)) \prec last(I_i)$.

Consider Case 1. Let $e^i_{i_1} = \mathcal{FCE}_i(last(I_j))$. There are two sub-cases: (a) $e^i_{i_1} \in S_i$. Therefore, $last(I_i) \prec last(S_i)$. This can happen only if there is some S_k such that $\mathcal{LCE}_i(last(S_k)) \preceq e^i_{i_1}$. (Note that if for all processes P_k $e^i_{i_1} \preceq \mathcal{LCE}_i(last(S_k))$, $e^i_{i_1} \preceq last(I_i)$.) This means that $\mathcal{LCE}_j(last(S_k)) \prec last(I_j)$ meaning that $last(I_j) \notin I_j$ (contradiction). (b) $e^i_{i_1} \notin S_i$. This means that $\mathcal{LCE}_j(last(S_i)) \prec last(I_j)$ meaning that $last(I_j) \notin I_j$ (contradiction).

Consider Case 2. This implies that $last(I_i) \prec \mathcal{FCE}_i(last(I_j))$ and the proof of Case 1 applies.

Cases 3 and 4 are similar to cases 1 and 2 respectively.

By a similar proof, we can show that $S_l = (first(I_1), \ldots, first(I_n))$ is a global state.■

Steps 3 and 4 of the algorithm of Figure 1 can be performed in $O(n^3)$ time and the other steps take $O(n)$ time. Thus, the time complexity is $O(n^3)$. A distributed implementation of the centralized algorithm is given in [8], and the message complexity is $O(n^3)$ where n is the total number of processes.

2.1.2 The General Case

In general, evaluation of a conjunct C_i on process P_i will give raise to multiple spectra where C_i holds. This situation can be handled by an extension of the technique just described.

Let S^i_1 denote the first spectrum at P_i where C_i holds. First, we apply the previous algorithm considering only the first spectrum at each process. If this results in a non-empty intersection of projections at every site, then we have found a state where Φ holds, and we terminate. Otherwise, at least one of the spectra is ignored and the algorithm is re-applied. Consider a situation when a state satisfying Φ cannot be found by projecting the first spectrum of the n processes. Thus there exists a process P_i such that the intersection I_i at P_i is empty. This may be the result of one of three causes.

1. There are two processes P_j and P_k such that $last(\pi(S_1^j \rightarrow P_i)) \prec first(\pi(S_1^k \rightarrow P_i))$. Then, it is clear that $last(S_1^j) \prec \mathcal{FCE}_j(first(S_1^k))$. Furthermore, there can be no spectrum S_l^k such that $\pi(S_l^k \rightarrow P_j)$ has a non-empty intersection with S_1^j. Therefore, we can ignore S_1^j from further consideration.

2. There is a process P_j such that $last(S_1^i) \prec first(\pi(S_1^j \rightarrow P_i))$. Similar to the previous case, it is clear that there cannot be a spectrum S_l^j such that $\pi(S_l^j \rightarrow P_i)$ has a non-empty intersection with S_1^i. We can, therefore, omit S_1^i from further consideration.

3. There is a process P_j such that $last(\pi(S_1^j \rightarrow P_i)) \prec first(S_1^i)$. This case is similar to Case 2. Obviously, we can omit S_1^j from further computations.

Several spectra may qualify to be candidates for removal. After removing such spectra, we consider the first "unremoved" spectra at each site and continue the process. Eventually, we reach a stage where we either exhaust all spectra at one or more processes thereby concluding that POSSIBLY(Φ) does not hold or we find a state satisfying Φ.

2.2 FIRST(Φ)

It may be observed that if there is a state in which Φ is true, the algorithm for POSSIBLY(Φ) will not only find that state but the state so found will also be FIRST(Φ).

2.3 LAST(Φ)

The algorithm for POSSIBLY(Φ) can be quite easily adapted to determine LAST(Φ). To do this, the spectra at each process are found as for POSSIBLY(Φ). However, projections of these spectra and the intersections of these projections are done in the reverse order. That is, we first consider the last spectrum at each site, project them, and take the intersections of these projections. If a non-empty intersection is obtained at each site, the last events of these intersections form LAST(Φ). If not, as we have shown for POSSIBLY(Φ), we will have at least one spectrum that can be ignored. (It is easy to show that there will be a process P_i with spectrum S_j^i such that $last(\pi(S_j^i \rightarrow P_k)) \prec first(S_l^k)$ where S_l^k is the last spectrum on P_k.)

2.4 DEFINITELY(Φ)

As in the case of the computation of POSSIBLY(Φ), we first address the problem of testing the property when the complete log is available. The algorithm is shown in Figure 2.

1. Evaluate C_i at each event of process P_i. For the moment, we will assume that Φ is such that the evaluation of each C_i results in at most one spectrum S_i.
2. If S_i is an empty spectrum, DEFINITELY(Φ) is false. Otherwise, apply the test below for all processes with non-empty spectra.
3. Test the log to see if there are processes P_i and P_j such that $e \not\prec last(S_i) \forall e \in S_j$ (that is, $last(S_i)$ is not causally dependent on any event in S_j). If such a pair of processes exists, DEFINITELY(Φ) is false. Otherwise, it is true.

Figure 2: Centralized algorithm for determining DEFINITELY(Φ)

Lemma 2 *Let the evaluation of Φ at each process result in exactly one non-empty spectrum S_k at each process P_k. Then, DEFINITELY(Φ) is false if and only if there exists a pair of processes P_i and P_j such that $first(S_j) \not\prec last(S_i)$.*

Proof. Assume that there is a process P_i such that for some process P_j, $j \neq i$, $last(S_i)$ is not causally dependent on $first(S_j)$. Let e^j_l be any event in S_j. Obviously, $last(S_i)$ is not causally dependent on e^j_l. Therefore, no event that precedes $last(S_i)$ can also be causally dependent on e^j_l. Then P_i can complete all of the events of S_i before P_j begins $first(S_j)$ since none of the events in S_i is dependent on any event in S_j. Thus we have constructed an execution where DEFINITELY(Φ) does not hold.

To prove necessity, assume that DEFINITELY(Φ) is false. Therefore, there exists an execution $\mathcal{E}$ in which Φ evaluates to false for all states. Let Σ_1 be the first state in this execution in which each process P_k has executed $first(S_k)$. Also, let Σ_2 be the first state in this execution in which some process P_l has executed the next event after $last(S_l)$. If $\Sigma_1 \prec \Sigma_2$, consider the state that is an immediate predecessor of Σ_2. All processes are in their respective spectrum, so Φ holds in $\mathcal{E}$ (contradiction). If $\Sigma_2 \prec Sigma_1$, P_l is able to complete $last(S_l)$ before some process P_k executes $first(S_k)$, proving our theorem. ■

2.4.1 The General Case

Suppose the evaluation of a conjunct at one or more sites generates more than one spectrum. In this case, the log is scanned backwards. Let $S^i_{i_m}$ denote the last spectrum at process P_i. Apply the algorithm described before. If this computation shows that DEFINITELY(Φ) holds, we are done. Otherwise, by Lemma 2, there is at least one pair of processes P_i and P_j such that $last(S^i_{i_m})$ does not causally depend on any event in $S^j_{j_m}$. It is then easy to see that for no spectrum $S^i_{i_k}$ on process P_i will $last(S^i_{i_k})$ be causally dependent on any event in $S^j_{j_m}$. Therefore, $S^j_{j_m}$ can be discarded from further consideration altogether. Instead, we consider "last" the spectrum $S^j_{j_l}$ such that $last(S^i_{i_m})$ causally depends on $first(S^j_{j_l})$. By "last", we mean that the property does not hold for any spectrum on P_j whose first element is a successor of $first(S^j_{jl})$. If for some P_k $last(S^k_{km})$ causally depends on some event in S^j_{jm}, $last(S^k_{km})$ will depend on all events in S^j_{jl} as well. All we need to compute is whether $last(S^j_{jl})$ causally depends on the spectra on the other $n-1$ processes.

3 Related Work

The works most related to this work is by Spezialetti [5], Spezialetti and Kearns [6], and Cooper and Marzullo [2]. The concepts of POSSIBLY(Φ), DEFINITELY(Φ), FIRST(Φ) and LAST(Φ) are from Cooper and Marzullo [2], and POSSIBLY(Φ) is similar to the *event occurrence* condition of [6]. The algorithms of Cooper and Marzullo [2] are centralized and may need an exponential amount of time (and an exponential number of messages if a distributed implementation is attempted) since their algorithm tests each possible global state. Our centralized techniques use $O(n^3)$ time in the worst case. Also, a distributed implementation of the techniques uses (n^3) messages in the worst case. (See [8] for details.) The algorithms presented in this paper find only the events within each process such that the global state represented by collecting all of these events satisfies the predicate. To restore the states of each of these processes just prior to these events or just after these events, the technique of reproducible execution of asynchronous distributed programs [7] may be used.

4 Conclusion and Future Work

The problem of detecting global predicates efficiently is considered and solutions have been presented in this paper. Algorithms for POSSIBLY(Φ), DEFINITELY(Φ), FIRST(Φ) and LAST(Φ) have been given. These techniques analyze a given "run" (or an execution) of a distributed program. Clearly, a single run of a distributed program creates an exponential number of global states, and it is possible to detect global predicates without explicitly considering each possible global state.

Systematically generating communication patterns one by one for completely testing the program (using our results) is the next step in testing distributed programs. There is no way to generate a finite number of distributed test cases which will "expose" all errors since the equivalent problem in sequential case is known to be undecidable. Different approaches such as (1) placing restrictions on the type of distributed programs, (2) probabilistic approach may be attempted. We are working on these and related problems.

References

[1] Chandy, K., and Lamport, L. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst. 3* (1985), 63–75.

[2] COOPER, R., AND MARZULLO, K. Consistent detection of global predicates. *SIGPLAN Notices* (1991), 167–174.

[3] FAGERSTROM, J. Design and test of distributed applications. In *Proceedings of the Tenth International Conference of Software Engineering* (1988), pp. 88–92.

[4] MACKERT, L., MEYER, R., SCHEERE, U., SCHNEIDER, J., VELTHUYS, R., BURMEISTER, J., MEER, J., AND SCHROER, I. A generalized conformance test tool for communication protocols. In *Proceedings of the Eighth International Conference on Distributed Computing Systems* (1988), pp. 180–188.

[5] SPEZIALETTI, M. *A generalized approach to monitoring distributed computations for event occurrences.* PhD thesis, University of Pittsburgh, Pittsburgh, Pennsylvania, 1989.

[6] SPEZIALETTI, M., AND KEARNS, P. Simultaneous regions: A framework for the consistent monitoring of distributed computations. In *Proceedings of the Ninth International Conference on Distributed Computing Systems* (1989), IEEE, pp. 61–68.

[7] VENKATESAN, S. Reproducible execution of distributed programs. Computer Science Technical Report UTDCS-8-91, The University of Texas at Dallas, 1991.

[8] VENKATESAN, S., AND DATHAN, B. Testing and debugging distributed programs using global predicates. Computer Science Technical Report UTDCS-11-92, The University of Texas at Dallas, June 1992.

[9] ZHANG. An interactive protocol synthesis algorithm using a global state transition graph. *IEEE Transactions on Software Engineering 14*, 3 (1988), 394–404.

Appendix

Procedure *pr_intersection* of Figure 3 gives the details of the centralized algorithm for finding projections and their intersections. The steps that revise $first(I_i)$ and $last(I_i)$ for all processes use $O(n^2)$ time per iteration and recomputing NMS^l and NMS^r takes $O(n^2)$ time per iteration. Since the total number of iterations is n, the time complexity of procedure *pr_intersection* is $O(n^3)$. The correctness of procedure *pr_intersection* is established below:

Theorem 1 *When procedure* pr_intersection *of Figure 3 terminates,* $I_i = \bigcap_{j=1}^{n} \pi(S_j \rightarrow P_i)$ *for all* $i = 1, \ldots, n$.

Proof. See [8].

```
procedure pr_intersection;
{ this procedure finds the intersection of projections of spectra }
begin
    for $i = 1$ to $n$ loop
        $I_i = S_i$;
    endloop;
    for each ordered pair of processes $(P_i, P_j)$ loop
        $\text{NMS}^l_{ij} \leftarrow$ number of messages sent by $P_i$ to $P_j$ till event $last(I_i)$;
        $\text{NMR}^l_{ij} \leftarrow$ number of messages received by $P_i$ from $P_j$ till event $last(I_i)$;
        $\text{NMS}^f_{ij} \leftarrow$ number of messages sent by $P_i$ to $P_j$ till event $first(I_i)$;
        $\text{NMR}^f_{ij} \leftarrow$ number of messages received by $P_i$ from $P_j$ till event $first(I_i)$;
    endloop;
    for ITERATIONS= 1 to $n$ loop
        for each ordered pair of processes $(P_i, P_j)$ loop
            if $\text{NMR}^l_{ij} > \text{NMS}^l_{ji}$ then
                { $last(I_i)$ cannot co-exist with $last(I_j)$ }
                $e \leftarrow$ latest event of $P_i$ such that $\text{NMS}^l_{ij}$ = the number of messages
                    received by $P_i$ from $P_j$ till $e$
                if $e$ occurs before $last(I_i)$ then $last(I_i) = e$;
                endif;
            endif;
            if $\text{NMS}^f_{ij} < \text{NMR}^f_{ji}$ then
                { $first(I_i)$ cannot co-exist with $first(I_j)$ }
                $e \leftarrow$ earliest event of $P_i$ such that $\text{NMR}^f_{ji}$ = the number of messages
                    sent by $P_i$ to $P_j$ till $e$;
                if $e$ occurs after $first(I_i)$ then $first(I_i) = e$;
                endif;
            endif;
        endloop; { inner for loop }
        if $last(I_i) \prec first(I_i)$ then
            $I_i = \phi$;
            stop;
        endif;
        recompute $\text{NMS}^l_{ij}, \text{NMR}^l_{ij}, \text{NMS}^f_{ij}, \text{NMR}^f_{ij}$ for all ordered pairs of
            processes $(P_i, P_j)$;
    endloop;
end;
```

Figure 3: procedure *pr_intersection*

Shared Processors versus Shared Networks [*]

Damas P. GRUSKA

Institute of Informatics, Comenius University
Mlynska dolina, 842 15 Bratislava, Czecho-Slovakia
gruska@mff.uniba.cs

Abstract

The time semantics of abstract programming languages is presented. It is based on an assumption that communicating processes do not share processors but that they may share communication network. It means that executions of actions cannot be interrupted during computations but that processes can wait for free communication links. The presented semantics is compared with the semantics based on an assumption that processes share processors.

1 Introduction

A notion of a communication can be considered as a common feature or an abstraction of different types of parallel computations and behaviour of distributed and concurrent systems. Starting from this paradigm many calculi or abstract programming languages are suggested to describe and to study these systems. They successfully model principles of behaviours of the systems as the "communicating systems". *Process Algebras* represent typical example of this approach. They allow us abstract many real properties of existing systems like durations and structure of their actions, properties of communication networks, distribution of system components and so on. Semantics for this calculi are defined in terms of equivalence relations on set of processes. Processes which cannot be distinguished by an observer are equivalent. So processes can be identified with corresponding equivalence classes.

Since for many applications these calculi are too abstract there is a need of either finer (more expressible) languages or finer semantics for existing calculi. A particular problem of standard Process Algebras is lack of tools for expressing durations of elementary actions. Their usual semantics are based on an assumption that concurrent computation can be viewed as an interleaving execution of atomic actions. For example, in this case processes $a.Nil \mid b.Nil$ and $a.b.Nil + b.a.Nil$ are equivalent.

But in case that we cannot neglect duration of these actions the interleaving semantics is completely inadequate. We need (time) semantics which are finer. Of course, durations of actions and other time properties strongly depend on an actual system architecture. Defining semantics which incorporate durations we cannot abstract such properties as a degree of parallelism and structure of interconnection networks. They are fundamental for timed behaviour of a system.

---

[*] Work supported by the grant MSMS SR 01/72

The aim of this paper is to develop a semantics for Process Algebras based on assumptions that concurrently running processes do not share processors (full parallelism) but that they can share an interconnection network. It is assumed that the network is of limited capacity and its momentary amount of free links depends on an amount and on locations of communicating processes. This might be the case of single bus, crossbar networks and so on (for more discussion on this point see [6]). It means that during an execution processes can wait for a free communication link. On this level of abstraction we neglect original reasons why there is no link at a moment. As a case study of Process Algebra we consider *A calculus of Communicating Systems* [8], CCS for short.

To define semantics for the time abstracting calculus CCS we introduce a timed calculus TiCCS which is based on the above mentioned assumption. Then starting from CCS process we express durations of its actions with the help of TiCCS calculus. We define two CCS processes equivalent if their timed versions behave in the same manner.

The main idea of TiCCS is to introduce into CCS t action. The execution of such a t action by a process indicates that the process is idling until the next clock cycle. We continue to use an intuitive assumption underlying the theory of CCS that all processes may idle indefinitely. Using the syntax of CCS the process $a.P$ can idle, i.e. it can perform a t action.

In our calculus we simulate durations of actions by postfixing them with an appropriate number of t actions. For example, let as consider CCS process $P = a.b.Nil$. If an action a has duration two time units and the next action b has duration three time units the resulting TiCCS process is equal to $P' = a.t.t.b.t.t.t.Nil$. This means that the action a can be performed anytime but the second atomic action b can be performed no earlier then after two ticks of a clock.

Semantics for CCS, defined as it was mentioned is not interleaving. For example, let us consider CCS processes $P = a.Nil \mid b.Nil$ and $Q = a.b.Nil + b.a.Nil$. Let us assume that durations of actions a and b are 1 time unit . We express this fact in calculus TiCCS and we get the following timed processes. $P' = a.t.Nil \mid b.t.Nil$ and $Q' = a.t.b.t.Nil + b.t.a.t.Nil$. Since P', Q' have different behaviour (only the first one can start b immediately after a) then also P, Q are distinguished.

Later we compare presented semantics with semantics suggested by Aceto and Hennessy [2]. Their approach is based on an assumption that processes share processors and so an execution of any action can be interrupted and the processor might be scheduled to another process for a while. Time for scheduling is neglected. So if an execution of an action takes some time and this execution is split to several phases then this time is given as a sum of durations of all these phases.

Roughly speaking their idea how to express duration of actions is to split every action to several phases, for simplicity, to two phases, its *starting* and its *finishing* phase (this idea is developed in [5]). Technically it is realized with the help of an *action refinement* (see [3, 4]) i.e. by the mapping $r : u \mapsto u_s.u_f$. Processes P, Q are defined to be *split equivalent* if $r(P), r(Q)$ are interleaving equivalent. Then processes $P = a.Nil \mid b.Nil$ and $Q = a.b.Nil + b.a.Nil$ are not split equivalent since processes $r(P) = a_s.a_f.Nil \mid b_s.b_f.Nil$ and $r(Q) = a_s.a_f.b_s.b_f.Nil + b_s.b_f.a_s.a_f.Nil$ are not interleaving equivalent. The first one can perform the action b_s immediately after a_s. The second one has to finish a before performing b_s.

The paper is organized as follows. In Section 2 we describe a language TiCCS and briefly recall CCS. Basic definitions and properties of TiCCS are presented in Section 3. In Sections 4 and 5 we introduce time and split semantics. Their relationship is discussed in Section 6.

In the interests of readability all proofs have been sketched or omitted from this paper. Almost all of them are rather technical and long.

2 The TiCCS language

To define the language TiCCS, we first presuppose a set of atomic action symbols A not containing symbols τ, t, and we define $Act = A \cup \{\tau\}, Actt = Act \cup \{t\}$. We let $a, b, \ldots$ range over A; $u, v, \ldots$ range over Act; and $x, y \ldots$ range over $Actt$.

The set of TiCCS terms is defined by the following BNF expression where we take $x \in Actt, X \in Var, S$ is ranging over relabelling functions $S : Actt \to Actt$ such that $\overline{S(a)} = S(\bar{a})$ for $a \in A, S(\tau) = \tau, S(t) = t$, and finally, $L \subseteq A$:

$$P ::= Nil \mid X \ \mid \ x.P \ \mid \ P + P \ \mid \ P \mid P \ \mid \ \mu XP \ \mid \ P \setminus L \ \mid \ P[S]$$

We use the usual definitions for free and bound variables, open and closed terms and guarded recursion. The set **TiCCS** of *processes* consists of closed and guarded TiCCS terms. The statement $P \xrightarrow{x} Q$ means that process P may evolve into process Q by performing the action x.

The operational semantics of TiCCS is defined in terms of Labelled Transition Systems, (LTS for short).

Definition 2.1 *A Labelled Transition System* is a triple $(S, T, \{\xrightarrow{x}, x \in T\})$ where S is a set of states, T is a set of labels and $\{\xrightarrow{x}, x \in T\}$ is the transition relation, such that each $\xrightarrow{x}$ is a binary transition relation on S. We write $P \xrightarrow{x} P'$ instead of $(P, P') \in \xrightarrow{x}$.

Definition 2.2 The transition relation $\{\xrightarrow{x}_{TiCCS}, x \in Actt\}$ is defined as the least relation satisfying the following inference rules:

$$\frac{}{x.P \xrightarrow{x} P} \qquad\qquad \frac{}{x.P \xrightarrow{t} x.P}$$

$$\frac{P \xrightarrow{u} P'}{P + Q \xrightarrow{u} P',\ Q + P \xrightarrow{u} P'} \qquad\qquad \frac{P \xrightarrow{t} P, Q \xrightarrow{t} Q'}{P + Q \xrightarrow{t} P' + Q'}$$

$$\frac{P \xrightarrow{u} P'}{P \mid Q \xrightarrow{u} P' \mid Q, \mid P \xrightarrow{u} Q \mid P'} \qquad\qquad \frac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P \mid Q \xrightarrow{t} P' \mid Q'}$$

$$\frac{P \xrightarrow{a} P', Q \xrightarrow{\bar{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q', Q \mid P \xrightarrow{\tau} Q' \mid P'} \qquad\qquad \frac{P \xrightarrow{x} P'}{P[S] \xrightarrow{S(x)} P'[S]}$$

$$\frac{P \xrightarrow{x} P'}{P \setminus L \xrightarrow{x} P' \setminus L}, (x, \bar{x} \notin L) \qquad\qquad \frac{P[\mu XP/X] \xrightarrow{x} P'}{\mu XP \xrightarrow{x} P'}$$

Our paradigm is expressed in the second rule for prefix operator and in the second rule for parallel composition. Since processes can wait for a free communication link we allow waiting to any action (also to τ). In case of parallel composition since an execution of both processes is independent in the sense that they do not share processors, the corresponding rule ensures that this execution cannot be interrupted.

The set **CCS** of CCS processes consists of closed and guarded TiCCS terms which do not contain the action t.

Remark. Every CCS process is also a TiCCS process.

Definition 2.3 The transition relation $\{\xrightarrow{x}_{CCS}, x \in Act\}$ is defined using the same inference rules as for TiCCS except the one for the action t.

Remark. If there is no danger of confusion we will not omit subscripts TiCCS and CCS.

3 Basic definitions and properties

The drawback of the semantics defined by the Definition 2.2 is that it is too intensional. Processes are differentiated even if no observer can distinguish them. The usual next step consists of defining a equivalence among processes. Then only its equivalence classes are considered. In this way equivalence relations define the semantics of CCS.

For a set S, let S^* to denote the set of finite sequences of elements of S. We denote the concatenation of sequences by juxtaposition. ϵ will denote the empty sequence.

Definition 3.1 Let $(\mathbf{TiCCS}, Actt, \rightarrow)$ be a labelled transition system (LTS). A relation $\Re \subseteq \mathbf{TiCCS} \times \mathbf{TiCCS}$ is called a *(strong) bisimulation* if it is symmetric and it satisfies the following condition: if $P \Re Q$ and $P \xrightarrow{x} P', x \in Actt$, then there exists a process Q' such that $Q \xrightarrow{x} Q'$ and $P' \Re Q'$.

Two processes P, Q are *(strongly) bisimilar*, abbreviated $P \sim Q$, if there exists a strong bisimulation relating P and Q.

It is easy to prove that the arbitrary union of strong bisimulations is again a strong bisimulation; $\sim$ is the maximal strong bisimulation on **TiCCS**. That is $\sim = \bigcup\{\Re, \Re$ is a strong bisimulation $\}$.

Theorem 3.1 $\sim$ is a congruence relation on the set of TiCCS expressions.

Proof. The proof is quite detailed and we omit it.

If $s = x_1 x_2 \ldots x_n, x_i \in Actt$, then we write

$$P \xrightarrow{s}_t Q \text{ instead of } P(\xrightarrow{t})^* \xrightarrow{x_1} (\xrightarrow{t})^* \ldots (\xrightarrow{t})^* \xrightarrow{x_n} (\xrightarrow{t})^* Q$$

The following definition formalizes in some sense strong bisimulation of CCS processes. According to the CCS calculus the fact that a process P may perform an action a and then behave like P' means that the process P *after some time* performs a and then *after some time* evolve into the process P'. In TiCCS we can express it by performing some sequences of t before and after a.

Definition 3.2 Let $(\mathbf{TiCCS}, Actt, \to)$ be a LTS. A relation $\Re \subseteq \mathbf{TiCCS} \times \mathbf{TiCCS}$ is called a *t-strong bisimulation* if it is symmetric and it satisfies the following condition: if $P \Re Q$ and $P \xrightarrow{k}_t P', k \in Act \cup \{\epsilon\}$, then there exists a process Q' such that $Q \xrightarrow{k}_t Q'$ and $P' \Re Q'$.

Two processes P, Q are *t-strongly bisimilar*, in short $P \sim_t Q$, if there exists a t-strong bisimulation relating P and Q.

It is easy to see that the arbitrary union of t-strong bisimulations is again a t-strong bisimulation; $\sim_t$ is the maximal t-strong bisimulation on **TiCCS**. That is $\sim_t = \bigcup\{\Re, \Re$ is a t-strong bisimulation $\}$.

Theorem 3.2 $\sim \subseteq \sim_t$.

Proof. It follows directly from the definitions that strong bisimulation is also a t-strong bisimulation.

Theorem 3.3 $\sim_t$ is a congruence relation on the set of TiCCS expressions.

Proof. Same as for a proof of the Theorem 3.1. Note that we have no problems with the $+$ operator as there are in case of weak bisimulation. Performing the t action (contrary to τ) the operator $+$ does not behave like a choice operator.

Finally we recall a definition of the strong bisimulation for CCS processes.

Definition 3.3 Let $(\mathbf{CCS}, Act, \to)$ be a labelled transition system (LTS). A relation $\Re \subseteq \mathbf{CCS} \times \mathbf{CCS}$ is called a *(strong) bisimulation* if it is symmetric and it satisfies the following condition: if $P \Re Q$ and $P \xrightarrow{u} P'$, then there exists a process Q' such that $Q \xrightarrow{u} Q'$ and $P' \Re Q'$.

Two processes P, Q are *(strongly) bisimilar*, abbreviated $P \sim Q$, if there exists a strong bisimulation relating P and Q.

Again it is to prove that the arbitrary union of strong bisimulations is again a strong bisimulation; $\sim$ is the maximal strong bisimulation on **CCS**. That is $\sim = \bigcup\{\Re, \Re$ is a strong bisimulation $\}$.

Remark. We use the same sign $\sim$ for the strong bisimulations on TiCCS and CCS, but they will be distinguished by the context.

4 Time semantics

To model duration of actions for CCS process we suffix every action a by the sequence of actions t where a number of t's is the duration of a. As a result we get a TiCCS process.

Definition 4.1 Let us consider a mapping $r : u \mapsto u.t^{i_u}, i \geq 0$ for every $u \in Act$. We define an extension of r on the set of TiCCS processes:

$$\begin{aligned} r(Nil) &= Nil \\ r(X) &= X \\ r(a.P) &= r(a).r(P) \\ r(P+Q) &= r(P)+r(Q) \\ r(P \mid Q) &= r(P) \mid r(Q) \\ r(P[S]) &= P[r \circ S] \\ r(\mu XP) &= \mu X r(P) \\ r(P \setminus L) &= r(P) \setminus L. \end{aligned}$$

We shall call this mapping a *time refinement*.

Example. Let as consider the process *Sender* which can read a message from an environment and then sends it to process *Receiver* which write it. (see Fig. 1) The CCS specification of the process Sending is as follows:

$$\begin{aligned} Sender &= read.\overline{send}.Nil \\ Receiver &= send.\overline{write}.Nil \\ Sending &= (Sender \mid Receiver) \setminus \{send\} \end{aligned}$$

If we know that the actions *read* takes three time units, *send* two and $\overline{send}, \overline{write}$ one time unit then the corresponding time refinement is as follows: $r(read) = read.t.t.t$, $r(send) = send.t.t, r(\overline{send}) = \overline{send}.t$ and $r(\overline{write}) = \overline{write}.t$. Then r(Sending) is the following process:

$$\begin{aligned} r(Sending) &= (r(Sender) \mid r(Receiver)) \setminus \{send\} \\ r(Sender) &= read.t.t.t.\overline{send}.t.Nil \\ r(Receiver) &= send.t.t.\overline{write}.t.Nil \end{aligned}$$

Theorem 4.1 Let $(\mathbf{TiCCS}, Actt, \rightarrow)$ be a LTS and $P \in \mathbf{CCS}$. Then $P \sim_t r(P)$, where r is an arbitrary time refinement.

Proof. From the definition of the t-strong bisimulation and the time refinement.

Corollary Let $P, Q \in \mathbf{CCS}$. Then $P \sim Q$ iff $r(P) \sim_t r(Q)$.

Proof. The proof is rather technical and we omit it.

Now we are ready to define a time bisimulation. Two processes in a calculus abstracting the time properties of actions (CCS) are equivalent if their more specified versions in the time model(TiCCS) are equivalent. For the sake of simplicity we restrict to the simple case when all actions take just one time unit but all results can be easily extended to arbitrary durations. Technically:

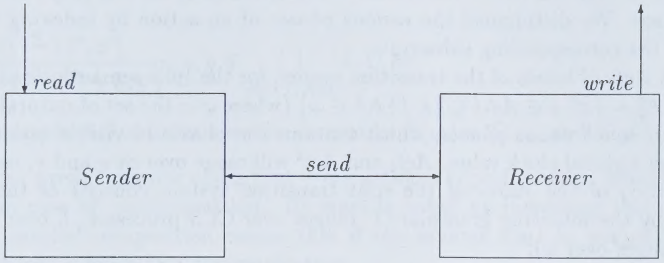


Figure 1: *Sending*

Definition 4.2 Two CCS processes P_1, P_2 are *time bisimilar* (denoted $\overset{t}{\sim}$) iff $r'(P_1) \sim r'(P_2)$, where r' is the time refinement such that $r' : u \mapsto u.t$ for every $u \in Act$.

Let $Sending' = read.\tau.\overline{write}.Nil$ is another specification of process of sending. It is easy to prove that $Sending \overset{t}{\sim} Sending'$.

Next Theorem claims that time bisimulation is the refinement or it is equal to strong bisimulation for CCS processes.

Theorem 4.2 Let $P, Q \in \mathbf{CCS}$. Then $P \overset{t}{\sim} Q$ implies $P \sim Q$.

Proof. Let $P \overset{t}{\sim} Q$. Then $r'(P) \sim r'(Q)$. According to the Theorems 3.2, 4.1 $P \sim_t r'(P) \sim_t r'(Q) \sim_t Q$, but by the previous Corollary $P \sim Q$.

An important property of time semantics is that it can be used for bottom-up specification techniques. With the help of the basic operators we can build more and more complicated processes starting from simple ones.

Theorem 4.3 $\overset{t}{\sim}$ is a congruence relation w.r.t. CCS syntax.

Proof. The proof is a direct consequence of the Theorem 3.1.

5 Split semantics

The definition of a split semantics is rather technical. It has to capture the requirement that when a process starts a synchronization with the first phase of an action (its starting), then the interaction must be completed by the following phase of the same action.

A way of implementing this requirement is to use a global clock whose elapsing is set by the beginning of actions. Hence a state will be a pair whose first component is a process and the second one is the current value of the global clock. The phases which must synchronize are postfixed by a superscript showing the clock value of the time at which the corresponding action started. Therefore, only phases postfixed by the same clock value can be synchronized (see [5] for details and proofs).

Let $Act^2 = Act^{(s)} \cup Act^{(f)}$ where $Act^{(i)} = \{u_i, u \in Act\}$ for $i \in \{s, f\}$ be the set of action phases. We distinguish the various phases of an action by indexing the action itself with the corresponding subscripts.

The set Act_2 of labels of the transition system for the split semantics will be $Act^2 \cup A_2^c$, where $A_2^c = \{a_i^k, a \in A \wedge i \in \{s, f\} \wedge k \in \omega\}$ (where ω is the set of natural numbers) is the set of *synchronous phases*, which contains the phases of visible actions in Act^2 postfixed by a global clock value. Act_2 and Act^2 will range over by η and ν, respectively.

The set S_2 of the states of the split transition system consists of the terms Q generated by the following grammar (P ranges over CCS processes, L over subsets of A and z ranges over ω):

$$\begin{aligned} R &::= P \mid \eta.P \mid R \,|\, R \mid R \setminus L \mid P[S] \\ Q &::= \langle R, z\rangle \mid \langle R \setminus A_2^c, z\rangle \end{aligned}$$

The initial state for any CCS process is $\langle P \setminus A_2^c, 1\rangle$.

Definition 5.1 The operational model for split semantics is labelled transition system $T = (S_2, Act_2, \{\xrightarrow{\eta}, \eta \in Act_2\})$ where the set of transitions is the least set of relations satisfying the following axioms and interface rules:

$$\frac{}{\langle u.P, z\rangle \xrightarrow{u_s} \langle u_f.P, z+1\rangle} \qquad \frac{}{\langle a.P, z\rangle \xrightarrow{a_s^z} \langle a_f^z.P, z+1\rangle}$$

$$\frac{}{\langle u_f.P, z\rangle \xrightarrow{u_f} \langle P, z\rangle} \qquad \frac{}{\langle a_f^k.P, z\rangle \xrightarrow{a_f^k} \langle P, z\rangle}$$

$$\frac{\langle P, z\rangle \xrightarrow{\eta} \langle P', z'\rangle}{\langle P+Q, z\rangle \xrightarrow{\eta} \langle P', z'\rangle} \qquad \frac{\langle P, z\rangle \xrightarrow{\eta} \langle P', z'\rangle}{\langle Q+P, z\rangle \xrightarrow{\eta} \langle P', z'\rangle}$$

$$\frac{\langle P, z\rangle \xrightarrow{\nu} \langle P', z'\rangle}{\langle P \mid Q, z\rangle \xrightarrow{\nu} \langle P' \mid Q, z'\rangle} \qquad \frac{\langle P, z\rangle \xrightarrow{a_i^k} \langle P', z'\rangle, \langle Q, z\rangle \not\xrightarrow{\bar{a}_i^k}}{\langle P \mid Q, z\rangle \xrightarrow{a_i^k} \langle P' \mid Q, z'\rangle}$$

$$\frac{\langle P, z\rangle \xrightarrow{\nu} \langle P', z'\rangle}{\langle Q \mid P, z\rangle \xrightarrow{\nu} \langle Q \mid P', z'\rangle} \qquad \frac{\langle P, z\rangle \xrightarrow{a_i^k} \langle P', z'\rangle, \langle Q, z\rangle \not\xrightarrow{\bar{a}_i^k}}{\langle Q \mid P, z\rangle \xrightarrow{a_i^k} \langle Q \mid P', z'\rangle}$$

$$\frac{\langle P, z\rangle \xrightarrow{a_i^k} \langle P', z'\rangle, \langle Q, z\rangle \xrightarrow{\bar{a}_i^k} \langle Q', z'\rangle}{\langle P \mid Q, z\rangle \xrightarrow{\tau_i} \langle P' \mid Q', z'\rangle} \qquad \frac{\langle P[\mu X P/X], z\rangle \xrightarrow{\eta} \langle P', z'\rangle}{\langle \mu X P, z\rangle \xrightarrow{\eta} \langle P', z'\rangle}$$

$$\frac{\langle P,z\rangle \xrightarrow{\eta} \langle P',z'\rangle}{\langle P[S],z\rangle \xrightarrow{S(\eta)} \langle P'[S],z'\rangle} \qquad \frac{\langle P,z\rangle \xrightarrow{\eta} \langle P',z'\rangle}{\langle P\setminus A_2^c,z\rangle \xrightarrow{\eta} \langle P'\setminus A_2^c,z'\rangle}, \eta \notin A_2^c$$

$$\frac{\langle P,z\rangle \xrightarrow{\eta} \langle P',z'\rangle}{\langle P\setminus L,z\rangle \xrightarrow{\eta} \langle P'\setminus L,z'\rangle}, \eta,\bar{\eta} \notin \bigcup_{i\in\{s,f\},k\in\omega} L_i \cup L_i^k$$

where the function S satisfies also the constrains $S(u_i) = S(u)_i$ and $S(u_i^k) = S(u)_i^k$.

Rules for prefix operator claim that every action a is performed in two phases a_s, a_f or a_s^z, a_f^z in case of communication. Its start is noted by increasing the clock value. Rules for parallel composition ensure that if two actions start to communicate then they also have to finish this communication.

Now we are ready to define a split semantics.

Definition 5.2 Two CCS processes P_1, P_2 are *split bisimilar* (denoted $\overset{s}{\sim}$) iff $\langle P_1 \setminus A_2^c, 1\rangle$ and $\langle P_2 \setminus A_2^c, 1\rangle$ are bisimilar in T.

Also the split bisimulation is the refinement of the strong bisimulation.

Theorem 5.1 Let $P, Q \in \mathbf{CCS}$. Then $P \overset{s}{\sim} Q$ implies $P \sim Q$.

Theorem 5.2 $\overset{s}{\sim}$ is a congruence relation w.r.t. the CCS syntax.

6 Time versus split semantics

The Theorems 4.2, 5.1 claim that both time and split semantics are finer or equal to strong bisimulation for CCS. In this section we prove that they are refinement of strong bisimulation and that they are different.

Theorem 6.1 $\overset{t}{\sim} \neq \sim$ and $\overset{s}{\sim} \neq \sim$, where $\sim$ is the strong bisimulation for CCS processes.

Proof. The processes $P = a.Nil \mid b.Nil, Q = a.b.Nil + b.a.Nil$ are strong equivalent but $P \overset{t}{\not\sim} Q, P \overset{s}{\not\sim} Q$.

As a first result on comparison of split and time semantics we have the following.

Theorem 6.2 $\overset{s}{\sim} \subseteq \overset{t}{\sim}$.

Proof. The proof is not straightforward since the both relations are defined with the help of the labelled transition systems with different sets of labels over transitions. Roughly speaking to show that one semantics is finer then other one we have to prove that there exists a function $f : A \to B$ such that $g_A = f \circ g_B$ where $g_A : S \to A$ and $g_B : S \to B$ are functions from language (a set of states) to domain of denotations (equivalence classes).

Next Theorem claims that time and split semantics are different. During a process specification we can choose between them. It depends on system architecture which one is adequate.

Theorem 6.3 $\overset{s}{\sim}\neq\overset{t}{\sim}$.

Proof. Let P be the process given by

$$P = (a.(\beta' \mid (\bar{\alpha}'.Nil + c.(\alpha.e.Nil + \gamma.P_1))) \mid b.(\bar{\alpha}.Nil \mid (\bar{\gamma}.Nil + c.(\beta.d.Nil + \alpha'.P_2)))) \setminus L$$

where $L = \{\alpha, \alpha', \beta, \gamma\}$, $P_1 = d.Nil \mid c.\tau.e.Nil$ and $P_2 = e.Nil \mid c.\tau.d.Nil$. Let Q be identical to P, but with the roles of d and e interchanged. (see [1]) It is easy to prove that $P \overset{t}{\sim} Q$ and $P \overset{s}{\not\sim} Q$. (The $a_s.a_f.c_s.b_s.b_f.c_f.\tau_s.\tau_f.e_s.e_f$ trace can be performed by P but not by Q.)

Conclusions

We presented time semantics for CCS. This semantics is sensitive to durations of actions and it is based on paradigm of sharing network and full parallelism. Presented semantics is can be used for bottom-up specification techniques. We compare this semantics with semantics based on paradigm of sharing processor. We showed that semantics are different.

References

[1] L. Aceto. Action Refinement in Process Algebra. TR 3/91, University of Sussex, Brighton, 1991.

[2] L. Aceto and M. Hennessy. Towards Action Refinement in Process Algebras. In *Proceedings of 4rd Annual IEEE Symposium on Logic in Computer Science*, Asilomar, California, pp. 138-145, 1989.

[3] R.J. van Glabbeek and U. Goltz. Equivalence Notions for Concurrent Systems and Refinement of Actions. In *MFCS'89*, LNCS 379, pp. 237-248, 1989.

[4] R. Gorrieri. Refinement, Atomicity and Transactions for Process Description Languages. Ph.D. Thesis, Report TD2/91, Dipartimento di Informatica, Università di Pisa, 1991.

[5] R. Gorrieri and C. Laneve. Split- and ST-Semantics for CCS. In *MFCS'91*, LNCS 520, pp. 170-180,1991.

[6] D.P. Gruska and A. Maggilo-Schettini. Process comunication environment. To be presented at North American Process Algebra Workshop, Stony Brook, 1992.

[7] H. Hansson and B. Jonsson. A Calculus for Communicating Systems with Time and Probabilities. In *Proceedings of 11th IEEE Real - Time Systems Symposium*, Orlando, 1990.

[8] R. Milner. *Communication and concurrency.* Prentice-Hall International, New York,1989.

The Complexity of Deciding Branching Bisimilarity for Normed Context-Free Processes

Didier Caucal
IRISA, Campus de Beaulieu
F35042 Rennes Cedex
France

Dung T. Huynh   *Lu Tian*
Computer Science Program
University of Texas at Dallas
Richardson, TX 75083

Abstract

We show that the problem of deciding branching bisimulation equivalence for normed context-free processes is in Σ_2^p, the second level of the polynomial-time hierarchy, and hence in **PSPACE**. We also show that minimization of normed context-free process graphs is in **PSPACE**.

1 Introduction

Complexity of process equivalences is an important research area in concurrency theories. It is known that all process equivalences are undecidable in any general algebraic theory of concurrent processes. Therefore considerable effort has been made to investigate interesting subclasses of processes for which process equivalences are decidable.

Bisimulation equivalence ([Pa-81]) is a very important one among process equivalences. Branching bisimulation equivalence ([GW-89]) is a more general version of bisimulation equivalence when processes under consideration have a silent action different from other observable actions. For finite processes, there are efficient polynomial time algorithms for deciding bisimulation equivalence ([KS-90] [PT-87]) and branching bisimulation equivalence ([GV-90]). Bisimulation equivalence for finite processes has been shown in [ABGS-91] to be even **P**-complete. Thus, branching bisimulation equivalence for finite processes is also **P**-complete.

An important step beyond finite processes was taken by Baeten, Bergstra and Klop who showed in [BBK-87] that bisimulation equivalence is decidable for normed context-free processes. Subsequently several simpler proofs for this decidability result were given in [Ca-90a], [HS-91] and [Gr-91]. Recently, Huynh and Tian showed in [HT-92] that bisimulation equivalence for normed context-free processes is in Σ_2^p, the second level of the polynomial-time hierarchy, and hence in **PSPACE**. Hüttel showed in [Hü-91a] that branching bisimulation equivalence is decidable for normed context-free processes.

In this paper, we study the complexity of branching bisimulation equivalence for normed context-free processes. It will be shown that the problem of deciding branching bisimulation equivalence for normed context-free processes is in Σ_2^p, the second level of the polynomial-time hierarchy, by applying the proof technique in [HT-92]. Hence, branching bisimulation equivalence for normed context-free processes is also in

PSPACE. Our proof, based on [HT-92], is simpler than the one in [Hü-91a]. Moreover, as the algorithm in [Hü-91a] appears to require at least exponential space, our complexity bound is considerably lower. The main idea of our algorithm is that two normed context-free processes are branching bisimilar iff there exists a certain polynomial bounded successful function whose closure maps the two processes to the process that is branching bisimilar to them. The dominant part of the algorithm is to verify the property of being a successful function. We will show that this verification is in $\mathbf{\Sigma_2^P}$ which yields the $\mathbf{\Sigma_2^P}$ upper bound for the entire algorithm as well. We also investigate the minimization problem for normed context-free process graphs which is the problem of computing for a given process graph an equivalent one that is minimal. The minimization problem for normed context-free process graphs was considered in [Ca-91]. In this paper, we show that this problem is in **PSPACE** as well.

In this work, **NL, P, NP**, *co*-**NP** and **PSPACE** have their standard meanings (cf. [GJ-79] [HU-79]). For k ≥ 1, a problem is said to be in $\mathbf{\Sigma_k^P}$ iff it is decidable in polynomial time by a nondeterministic Turing machine with a $\mathbf{\Sigma_{k-1}^P}$ oracle, where $\mathbf{\Sigma_0^P} = \mathbf{P}$ (cf. [St-76]). *co*-$\mathbf{\Sigma_k^P}$ is denoted by $\mathbf{\Pi_k^P}$. It is well-known that

$$\mathbf{NL} \subseteq \mathbf{P} \subseteq \begin{array}{c} \mathbf{NP} = \mathbf{\Sigma_1^P} \\ co\text{-}\mathbf{NP} = \mathbf{\Pi_1^P} \end{array} \subseteq \begin{array}{c} \mathbf{\Sigma_2^P} \\ \mathbf{\Pi_2^P} \end{array} \subseteq \cdots \subseteq \mathbf{PSPACE}.$$

This paper is organized as follows. In Section 2, we introduce some basic definitions and facts. Section 3 contains the main technical result for achieving the $\mathbf{\Sigma_2^P}$ upper bound for branching bisimulation equivalence of normed context-free processes. In Section 4 we derive the upper bound. In Section 5 we show that minimization of normed context-free process graphs is in polynomial space. Because of space limitation we omit all proofs which can be found in a technical report available from the authors.

2 Basic Definitions and Facts

In this section, we introduce the notion of normed context-free processes with silent transitions and the notion of branching bisimulation equivalence. Moreover, we state some basic facts that will be used later on.

Definition 2.1 A *process graph* M is a 4-tuple $M = (P, Act, \delta, p_0)$ where

1. P is a set of *processes.*
2. Act is a finite set of *actions.* A special action $\tau \in Act$ is called the *silent action.*
3. $\delta \subseteq P \times Act \times P$ is the *transition relation.*
4. $p_0 \in P$ is the *initial process.*

We say that there is a transition labeled by action a from process p to process q, denoted by $p \to^a q$, if $(p, a, q) \in \delta$. We write $p \to^a$ (resp. $p \to q$) if $p \to^a q$ for some process q (resp. action a). If

$$p(\to^\tau)^* \to^{a_1} (\to^\tau)^* \to^{a_2} \cdots \to^{a_k} (\to^\tau)^* q$$

where $a_i \neq \tau$ $(i = 1, 2, \ldots, k)$, then we call this a *(transition) path* labeled by action string $a_1 a_2 \cdots a_k$ from process p to process q. We write $p \Rightarrow^{a_1 \cdots a_k} q$ if such a path exists. In particular, $p \Rightarrow^{\epsilon} q$ means that $p(\rightarrow^{\tau})^* q$.

We now define the process graphs specified by context-free grammars in Greibach normal form.

Definition 2.2 Let $G = (Var, Act, \Pi, S)$ be a context-free grammar (CFG), where Var is the set of nonterminals, Act is the set of terminals, Π is the set of productions and S is the start symbol in Var. G is said to be in *Greibach normal form (GNF)* if each production in Π is of the form $X \rightarrow a\alpha$, where a is a terminal in Act and α is a nonterminal string in Var^*. Further, G is said to be in m-GNF if the right hand side of each production in Π contains at most m nonterminals.

Every GNF CFG may be regarded as a (weakly) guarded system of recursion equations over the basic process algebra (BPA) (cf. [BBK-87]). Therefore, every GNF CFG can be associated with a process graph as follows.

Definition 2.3 Let $G = (Var, Act, \Pi, S)$ be a GNF CFG. The *(context-free) process graph* M_G specified by G is $M_G = (Var^*, Act, \delta, \sigma)$ where:

1. Every $\alpha \in Var^*$ is called a *(context-free) process*,
2. A distinguished symbol in Act is specified as the silent action τ,
3. $\delta := \{\, X\beta \rightarrow^{a} \alpha\beta \mid X \rightarrow a\alpha \in \Pi,\ X \in Var,\ a \in Act,\ \textit{and}\ \ \alpha, \beta \in Var^* \,\}$,
4. A (context-free) process $\sigma \in Var^*$ is specified as the *initial (context-free) process.*

Note that in defining M_G we allow any $\sigma \in Var^*$ to be the initial process of M_G. For a string s, $length(s)$ denotes the number of symbols in s. The *norm* of a process $\alpha \in Var^*$ is defined to be

$$norm(\alpha) := min[\{\, length(x) \mid \alpha \Rightarrow^{x} \epsilon \ and\ x \in (Act - \{\tau\})^* \} \cup \{+\infty\}].$$

We say that G is *(weakly) normed* ([Hü-91a]) if for all $X \in Var$, $0 < norm(X) < +\infty$. In this case, every $\alpha \in Var^*$ is called a *(weakly) normed context-free process.*

Definition 2.4 Let $M = (P, Act, \delta, p_0)$ be a process graph. A relation $R \subseteq P \times P$ is said to be a *branching bisimulation* for M if for all $(p, q) \in R$ and $a \in Act$,

1. $p \rightarrow^{a} p'$ implies that either
    - $a = \tau$ and $(p', q) \in R$, or
    - $\exists\, q'_0, q'_1, \ldots, q'_k, q' :\ q = q'_0 \rightarrow^{\tau} q'_1 \rightarrow^{\tau} \cdots \rightarrow^{\tau} q'_k \rightarrow^{a} q'$ with $(p, q'_i), (p', q') \in R$ $(0 \leq i \leq k)$.
2. $q \rightarrow^{a} q'$ implies that either
    - $a = \tau$ and $(p, q') \in R$, or
    - $\exists\, p'_0, p'_1, \ldots, p'_k, p' :\ p = p'_0 \rightarrow^{\tau} p'_1 \rightarrow^{\tau} \cdots \rightarrow^{\tau} p'_k \rightarrow^{a} p'$ with $(p'_i, q), (p', q') \in R$ $(0 \leq i \leq k)$.

Two processes p and q are said to be *branching bisimulation equivalent* (or *branching bisimilar*), denoted by $p \sim_{bb} q$, if there is a branching bisimulation R for M with $(p, q) \in R$.

For two process graphs M and N, one can define the notion of *branching bisimulation* between M and N in a similar way. M and N are said to be *branching bisimulation equivalent* (or *branching bisimilar*) if there is a branching bisimulation between them with the initial processes being related.

We note that the notion of branching bisimulation equivalence was introduced in [GW-89]. It is known that branching bisimulation equivalence is the maximal branching bisimulation. The following characterization of branching bisimulation was shown in [Gl-90].

Proposition 2.5 *Let $M = (P, Act, \delta, p_0)$ be a process graph and $R \subseteq P \times P$. Then R is a branching bisimulation iff for all $(p, q) \in R$ and $a \in Act$,*

1. $p \to^a p'$ *implies that either*
   - $a = \tau$ *and* $(p', q) \in R$, *or*
   - $\exists\, q_1', q' :\ q \Rightarrow^\epsilon q_1' \to^a q'$ *with* $(p, q_1'), (p', q') \in R$.
2. $q \to^a q'$ *implies that either*
   - $a = \tau$ *and* $(p, q') \in R$, *or*
   - $\exists\, p_1', p' :\ p \Rightarrow^\epsilon p_1' \to^a p'$ *with* $(p_1', q), (p', q') \in R$.

The following proposition shows the similarity between branching bisimulation equivalence and bisimulation equivalence, and is the principle of our decision algorithm for branching bisimulation equivalence for normed context-free processes.

Proposition 2.6 *Let $M = (P, Act, \delta, p_0)$ be a process graph and $p, q \in P$. Then $p \sim_{bb} q$ iff for all $a \in Act$,*

1. $p \Rightarrow^\epsilon p_1 \to^a p_2$ *and* $p \sim_{bb} p_1$ *imply that either*
   - $a = \tau$ *and* $p_1 \sim_{bb} p_2$, *or*
   - $\exists\, q_1, q_2 :\ q \Rightarrow^\epsilon q_1 \to^a q_2$ *with* $p_1 \sim_{bb} q_1$ *and* $p_2 \sim_{bb} q_2$.
2. $q \Rightarrow^\epsilon q_1 \to^a q_2$ *and* $q \sim_{bb} q_1$ *imply that either*
   - $a = \tau$ *and* $q_1 \sim_{bb} q_2$, *or*
   - $\exists\, p_1, p_2 :\ p \Rightarrow^\epsilon p_1 \to^a p_2$ *with* $p_1 \sim_{bb} q_1$ *and* $p_2 \sim_{bb} q_2$.

Lemma 2.7 *Let $M = (P, Act, \delta, p_0)$ be a process graph and $p, q, r \in P$. If $p \Rightarrow^\epsilon q \Rightarrow^\epsilon r$ and $p \sim_{bb} r$, then $p \sim_{bb} q$.*

Lemma 2.8 [Hü-91b] *Branching bisimulation equivalence is a congruence relation for context-free processes.*

The following "split" lemma was shown in [Ca-91], which is a more general version of the "split" lemma in [HT-92]. Their proofs are similar.

Lemma 2.9 *Let $G = (Var, Act, \Pi, S)$ be a normed GNF CFG. Then, for all α, α', β and $\beta' \in Var^*$, $\alpha\beta \sim_{bb} \alpha'\beta'$ and $norm(\alpha) \leq norm(\alpha')$ iff $\alpha\xi \sim_{bb} \alpha'$ and $\beta \sim_{bb} \xi\beta'$ for some $\xi \in Var^*$.*

Corollary 2.10 *Let $G = (Var, Act, \Pi, S)$ be a normed GNF CFG. Then, for all α, α', β and $\beta' \in Var^*$, $\alpha\beta \sim_{bb} \alpha'\beta'$ and $norm(\alpha) = norm(\alpha')$ iff $\alpha \sim_{bb} \alpha'$ and $\beta \sim_{bb} \beta'$.*

We end this section by showing the following useful lemma.

Lemma 2.11 *Let $G = (Var, Act, \Pi, S)$ be a normed GNF CFG and $\alpha, \beta \in Var^*$. The problem of deciding whether $\alpha \Rightarrow^{\epsilon} \beta$ is* **NL***-complete.*

3 Fundamental Result

In this section we show some technical results concerning branching bisimulation equivalence for normed context-free processes. These results allow us to derive a $\mathbf{\Sigma_2^P}$ algorithm for deciding branching bisimilarity of normed context-free processes. In the following let $G = (Var, Act, \Pi, S)$ be a normed GNF CFG. $X, Y, \ldots$ range over Var and $\alpha, \beta, \ldots$ range over Var^*.

Let $g : Var \longrightarrow Var^+$ be a function. We extend g to a homomorphism $g : Var^* \longrightarrow Var^*$ by letting $g(\epsilon) := \epsilon$ and $g(X\alpha) := g(X)g(\alpha)$. Moreover, we let $g^0(X) := X$ and $g^{i+1}(X) := g(g^i(X))$. We say that g is *norm-preserving* if, for all $X \in Var$, $norm(g(X)) = norm(X)$.

Definition 3.1 Let $g : Var \longrightarrow Var^+$ be a function. g is said to be *eventually idempotent* if $g^i = g^{i+1}$ for some integer $i \geq 0$. In this case, we write $g^* = g^i$. g is said to be *idempotent* if $g^2 = g$.

The notions of idempotent functions and norm-preserving functions were introduced in [Gr-91], while eventually idempotent functions were defined in [HT-92].

Lemma 3.2 *Let $g : Var \longrightarrow Var^+$ be eventually idempotent. Then g^* is idempotent. Moreover, if g is norm-preserving, then so is g^*.*

Definition 3.3 An eventually idempotent function $g : Var \longrightarrow Var^+$ is said to be *successful* (with respect to $\sim_{bb}$) iff for all $X \in Var$,

1. $X \Rightarrow^{\epsilon} \alpha_1 \rightarrow^{a} \alpha_2$ and $g^*(X) = g^*(\alpha_1)$ imply that either
   - $a = \tau$ and $g^*(X) = g^*(\alpha_2)$, or
   - $\exists\, \beta_1,\, \beta_2 :\ g(X) \Rightarrow^{\epsilon} \beta_1 \rightarrow^{a} \beta_2$ with $g^*(\alpha_1) = g^*(\beta_1)$ and $g^*(\alpha_2) = g^*(\beta_2)$.
2. $g(X) \Rightarrow^{\epsilon} \beta_1 \rightarrow^{a} \beta_2$ and $g^*(X) = g^*(\beta_1)$ imply that either
   - $a = \tau$ and $g^*(X) = g^*(\beta_2)$, or
   - $\exists\, \alpha_1,\, \alpha_2 :\ X \Rightarrow^{\epsilon} \alpha_1 \rightarrow^{a} \alpha_2$ with $g^*(\alpha_1) = g^*(\beta_1)$ and $g^*(\alpha_2) = g^*(\beta_2)$.

We note that the notion of a successful function is closely related to the notion of a *self-proving relation* defined in [Ca-91]. The following lemma characterizes the property of being successful of an eventually idempotent function g.

Lemma 3.4 *Let* $g : Var \rightarrow Var^+$ *be an eventually idempotent function. Then* g *is successful iff for all* $\alpha \in Var^*$,

1. $\alpha \Rightarrow^{\epsilon} \alpha_1 \rightarrow^{a} \alpha_2$ *and* $g^*(\alpha) = g^*(\alpha_1)$ *imply that either*
   - $a = \tau$ *and* $g^*(\alpha) = g^*(\alpha_2)$, *or*
   - $\exists\ \beta_1,\ \beta_2 :\ g^*(\alpha) \Rightarrow^{\epsilon} \beta_1 \rightarrow^{a} \beta_2$ *with* $g^*(\alpha_1) = g^*(\beta_1)$ *and* $g^*(\alpha_2) = g^*(\beta_2)$.
2. $g^*(\alpha) \Rightarrow^{\epsilon} \beta_1 \rightarrow^{a} \beta_2$ *and* $g^*(\alpha) = g^*(\beta_1)$ *imply that either*
   - $a = \tau$ *and* $g^*(\alpha) = g^*(\beta_2)$, *or*
   - $\exists\ \alpha_1,\ \alpha_2 :\ \alpha \Rightarrow^{\epsilon} \alpha_1 \rightarrow^{a} \alpha_2$ *with* $g^*(\alpha_1) = g^*(\beta_1)$ *and* $g^*(\alpha_2) = g^*(\beta_2)$.

We are now ready to state our key result.

Theorem 3.5 *For all* $\alpha, \beta \in Var^*$, $\alpha \sim_{bb} \beta$ *iff* $g^*(\alpha) = g^*(\beta)$ *for some successful function* $g : Var \rightarrow Var^+$.

We provide a proof sketch for Theorem 3.5 below. The "if" part is done by

Proposition 3.6 *Let* $g : Var \rightarrow Var^+$ *be a successful function. Then*

$$R = \{(\alpha, \beta) \in Var^* \times Var^* \mid g^*(\alpha) = g^*(\beta)\}$$

is a branching bisimulation.

The proof of the "only-if" part consists of the following definition and lemma. We assume a total ordering (commonly called the standard ordering) $<$ on Var^* as follows:

$$\alpha < \beta \ \text{ iff} \begin{cases} length(\alpha) < length(\beta) \ \ \text{or} \\ \alpha \text{ is lexicographically smaller than } \beta \text{ and } length(\alpha) = length(\beta). \end{cases}$$

Definition 3.7 The function $g_{min} : Var \rightarrow Var^+$ is defined by:

$$g_{min}(X) = \begin{cases} min\{\alpha \in Var^+ \mid X \sim_{bb} \alpha \text{ and } X < \alpha\} & \text{if any} \\ X \quad \text{otherwise} \end{cases}$$

Lemma 3.8 *Let* $g_{min} : Var \rightarrow Var^+$ *be as defined in Definition 3.7. Then for all* $\alpha \in Var^*$,

1. $g_{min}(\alpha) = \begin{cases} \text{some } \gamma \text{ such that } \alpha \sim_{bb} \gamma \text{ and } \alpha < \gamma & \text{if any} \\ \alpha \quad \text{otherwise} \end{cases}$
2. g_{min} *is a successful function.*
3. $g^*_{min}(\alpha) = max\{\gamma \mid \gamma \sim_{bb} \alpha\}$.

We finish this section with the following

Corollary 3.9 *Every successful function* $g : Var \rightarrow Var^+$ *is norm-preserving.*

4 Complexity Upper Bound

In this section we show that Theorem 3.5 yields a $\mathbf{\Sigma_2^P}$ upper bound for deciding branching bisimilarity for normed context-free processes. Assume without loss of generality that we are given a normed m-GNF CFG $G = (Var, Act, \Pi, S)$ that contains n nonterminals in Var.

Definition 4.1 A transition path in M_G of the form $\alpha_0 \to \alpha_1 \to \alpha_2 \to \ldots \to \alpha_t$ is said to be a *norm-non-increasing path* if $norm(\alpha_{i+1}) \leq norm(\alpha_i)$ for all $0 \leq i < t$.

Proposition 4.2 *Let $\gamma \in Var^*$ be a process on a norm-non-increasing path beginning with process $\alpha \in Var^*$. Then $length(\gamma) \leq length(\alpha) + (n-1)(m-1)$.*

Therefore, we can restate Theorem 3.5 as

Corollary 4.3 *For all $\alpha, \beta \in Var^*$, $\alpha \sim_{bb} \beta$ iff $g^*(\alpha) = g^*(\beta)$ for some successful function $g : Var \to Var^{\leq (n-1)(m-1)+2}$.*

Proposition 4.4 [HT-92]

- *Let $g : Var \to Var^+$ be a function. Then the problem of deciding whether g is eventually idempotent is in* **P**.
- *Let $g : Var \to Var^+$ be an eventually idempotent function and $\alpha, \beta \in Var^*$. Then the problem of deciding whether $g^*(\alpha) = g^*(\beta)$ is in co-***NP**.

From Lemma 2.11 and Proposition 4.4, one can easily see that verifying whether an eventually idempotent function is successful is in $\mathbf{\Pi_3^P}$ which yields a $\mathbf{\Sigma_4^P}$ algorithm for branching bisimulation. We now show how to improve this bound to $\mathbf{\Sigma_2^P}$.

For $\alpha, \beta \in Var^*$, we write $\alpha \leq_{pre} \beta$ if α is a prefix of β. If $length(\alpha) \leq length(\beta)$, the *quotient* of β by α is defined by $\beta/\alpha := \gamma$ such that $\beta = \xi\gamma$ for some $\xi \in Var^*$ with $length(\xi) = length(\alpha)$. The following characterization of a successful function is important in verifying in $\mathbf{\Sigma_2^P}$ whether an eventually idempotent function is successful.

Proposition 4.5 *Let $g : Var \to Var^*$ be an eventually idempotent function. Then g is successful iff for all $X, Y \in Var$, $X \to^a \alpha$ implies that either $a = \tau$ and $g^*(X) = g^*(\alpha)$, or*

1. - $g^*(Y) \leq_{pre} g^*(X)$ *and*
   - $\exists \beta_1, \beta_2 : Y \Rightarrow^\epsilon \beta_1 \to^a \beta_2$ *with* $g^*(Y) = g^*(\beta_1)$, $g^*(\beta_2) \leq_{pre} g^*(\alpha)$ *and* $g^*(X)/g^*(Y) = g^*(\alpha)/g^*(\beta_2)$, *or*
2. - $g^*(X) \leq_{pre} g^*(Y)$ *and*
   - $\exists \beta_1, \beta_2 : Y \Rightarrow^\epsilon \beta_1 \to^a \beta_2$ *with* $g^*(Y) = g^*(\beta_1)$, $g^*(\alpha) \leq_{pre} g^*(\beta_2)$ *and* $g^*(Y)/g^*(X) = g^*(\beta_2)/g^*(\alpha)$.

Proposition 4.6 *Let $g : Var \to Var^+$ be an eventually idempotent function. Then the problem of deciding whether g is successful is in $\mathbf{\Sigma_2^P}$.*

From the above propositions, the $\mathbf{\Sigma_2^P}$ upper bound of branching bisimulation equivalence for normed context-free processes follows.

Theorem 4.7 *The problem of deciding branching bisimulation equivalence for normed context-free processes is in $\mathbf{\Sigma_2^P}$.*

5 Minimizing Normed Context-Free Processes

In this section, we show that, given a normed context-free grammar, one can construct in polynomial space a normed context-free grammar which specifies the minimal normed context-free process graph branching bisimilar to the one specified by the original grammar. We note that the minimal process graph under branching bisimulation equivalence is unique up to isomorphism.

Definition 5.1 Let $M = (P, Act, \delta, p_0)$ be a process graph. The *quotient* $M/\sim_{bb}$ of M by $\sim_{bb}$ is the process graph $M/\sim_{bb} := (P', Act, \delta', [p_0])$ where:

1. for all $p \in P$, $[p] := \{q \in P \mid p \sim_{bb} q\}$, i.e. $[p]$ is the $\sim_{bb}$ equivalence class of p.
2. $P' := \{[p] \mid p \in P\}$.
3. $\delta' := \{[p] \rightarrow^a [q] \mid (p \rightarrow^a q) \in \delta$ *and if* $[p] = [q]$ *then* $a \neq \tau\}$.

In orther words, $M/\sim_{bb}$ is the process graph obtained by identifying equivalent processes and then removing τ-loops ([Ca-91]).

Proposition 5.2 *Let* $M = (P, Act, \delta, p_0)$ *be a process graph. Then* $M \sim_{bb} M/\sim_{bb}$.

Definition 5.3 Let $M = (P, Act, \delta, p_0)$ and $N = (Q, Act, \theta, q_0)$ be two process graphs. M and N are said to be *isomorphic*, denoted by $M \sim_{iso} N$, if there exists a bijection $f : P \rightarrow Q$ such that $f(p_0) = q_0$, and for all $p, p' \in P$ and $a \in Act$, $p \rightarrow^a p'$ iff $f(p) \rightarrow^a f(p')$.

Definition 5.4 A process graph M is said to be *minimal* (with respect to $\sim_{bb}$) if it is isomorphic to its quotient $M/\sim_{bb}$.

Now assume that $G = (Var, Act, \Pi, S)$ is a normed m-GNF CFG ($m \geq 2$), $\sigma \in Var^*$ is the initial process in M_G and n is the number of nonterminals in Var. Recall that (see Lemma 3.8): $g^*_{min}(\alpha) = max\{\gamma \mid \gamma \sim_{bb} \alpha\}$.

Proposition 5.5 *Let* $M = (P, Act, \delta, p_0)$ *be the process graph such that*

1. $P := \{g^*_{min}(\alpha) \mid \alpha \in Var^*\}$.
2. $p_0 := g^*_{min}(\sigma)$.
3. $\delta := \{g^*_{min}(\alpha) \rightarrow^a g^*_{min}(\beta) \mid \alpha \rightarrow^a \beta$ *in* M_G *and if* $g^*_{min}(\alpha) = g^*_{min}(\beta)$ *then* $a \neq \tau\}$.

Then $M \sim_{iso} M_G/\sim_{bb}$.

From the above proposition it is not hard to prove the following

Theorem 5.6 *Minimization of normed context-free process graphs is in* **PSPACE**.

References

[ABGS-91] C. Alvarez, J. L. Balcázar, J. Gabarró and M. Santha. Parallel Complexity in the Design and Analysis of Concurrent Systems. *Lecture Notes in Computer Science 505*, pp. 288-303, 1991.

[BBK-87] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages. *Lecture Notes in Computer Science 259*, pp. 94-113, 1987.

[Ca-90a] D. Caucal. Graphes Canoniques de Graphes Algébriques. *Theoretical Informatics and Applications*, 24(4), pp. 339-352, 1990.

[Ca-90b] D. Caucal. On the Regular Structures of Prefix Rewriting. *Lecture Notes in Computer Science 431*, pp. 87-102, 1990.

[Ca-91] D. Caucal. Branching Bisimulation for Context-Free Processes, Unpublished Manuscript, 1991.

[GJ-79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York, 1979.

[Gl-90] R. J. van Glabbeek. Comparative Concurrency Semantics and refinement of Actions. Ph.D. Thesis, CWI/Vrije Universiteit, 1990.

[Gr-91] J. F. Groote. A Short Proof of the Decidability of Bisimulation for Normed BPA-Processes. Technical Report CS-R9151, CWI, 1991. To appear in *Information Processing Letters.*

[GV-90] J. F. Groote and F. Vaandrager. An Efficient Algorithm for Branching Bisimulation and Stuttering Equivalence. *Lecture Notes in Computer Science 443*, pp. 626-638, 1990.

[GW-89] R. J. van Glabbeek and W. P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Information Processing 89*, pp. 613-618, 1989.

[HS-91] H. Hüttel and C. Stirling. Actions Speak Louder than Words: Proving Bisimilarity for Context-Free Processes. *Proc. 6th Annual Symp. on Logic in Computer Science*, pp. 376-386, 1991.

[HT-91] D. T. Huynh and L. Tian. Complexity of Deciding Readiness and Failure Equivalences for Processes. *Proc. of the 3rd IEEE Symp. on Parallel and Distributed Processing*, pp. 738-745, 1991.

[HT-92] D. T. Huynh and L. Tian. Deciding Bisimilarity of Normed Context-Free Processes Is in Σ_2^p. Technical Report UTDCS-1-92, University of Texas at Dallas, 1992. To appear in *Theoretical Computer Science.*

[HU-79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Co., 1979.

[Hü-91a] H. Hüttel. Silence is Golden: Branching Bisimilarity is Decidable for Context-Free Processes. *Proc. of CAV91.* Technical Report ECS-LFCS-91-173, University of Edinburgh, 1991.

[Hü-91b] H. Hüttel. Decidability, Behavioural Equivalences and Infinite Transition Graphs. Ph.D. Thesis, University of Edinburgh, 1991.

[KS-90] P. C. Kanellakis and S. A. Smolka. CCS Expressions, Finite State Processes and Three Problems of Equivalence. *Information and Computation 86*, pp. 43-68, 1990.

[Mi-89] R. Milner. *Communication and Concurrency*, Prentice-Hall, 1989.

[Pa-81] D. Park. Concurrency and Automata on Infinite Sequences. *Lecture Notes in Computer Science 104*, pp. 168-183, 1981.

[PT-87] R. Paige and R. Tarjan. Three Partition Refinement Algorithms. *SIAM J. on Computing 16*, pp. 973-989, 1987.

[St-76] L. J. Stockmeyer. The Polynomial Time Hierarchy. *Theoretical Computer Science 3*, pp. 1-22, 1976.

MPSK ERROR BOUNDS USING ALTERNATING FUNCTIONS

John J. Komo and Kevin D. Barnett
Electrical and Computer Engineering
Clemson University
Clemson, SC 29634

ABSTRACT

A technique for bounding integrals using alternating functions is presented here. Since the difference of the original function and the bounding function is not strictly of the same sign, as used in previous bounding techniques, tighter bounds can be obtained. As an application these bounding techniques are used to upper and lower bound MPSK word error probabilities.

INTEGRAL BOUNDS

For f(x) a positive function and g(x) a monotonic non-increasing positive function it is desired to upper and lower bound the integral

$$I = \int_a^b g(x)f(x)dx. \tag{1}$$

The integral can clearly be bounded by

$$\int_a^b g(x)h_2(x)dx \le \int_a^b g(x)f(x)dx = I \le \int_a^b g(x)h_1(x)dx \tag{2}$$

where $h_2(x) \le f(x) \le h_1(x)$ for $a \le x \le b$.

Consider the case where for some c_1 on the interval [a,b], $f(x) \le h_1(x)$ for $a \le x \le c_1$ and $f(x) \ge h_1(x)$ for $c_1 \le x \le b$. Thus the difference $f(x)-h_1(x)$ changes sign once on the interval [a,b], rather than being strictly nonpositive as used in the upper bound of (2). Also, consider the case where for some c_2 on the interval [a,b], $f(x) \ge h_2(x)$ for $a \le x \le c_2$ and $f(x) \le h_2(x)$ for $c_2 \le x \le b$. The difference $f(x)-h_2(x)$ also changes sign once on the interval [a,b], rather than being strictly nonnegative as used in the lower bound of (2).

The integral I can be written

$$I = \int_a^b g(x)h_i(x)dx + \int_a^{c_i} g(x)[f(x)-h_i(x)]dx + \int_{c_i}^b g(x)[f(x)-h_i(x)]dx. \tag{3}$$

Since $f(x)-h_1(x) \le 0$, $a \le x \le c_1$, the integral from a to c_1 can be upper bounded by replacing g(x) by its minimum value on $[a,c_1]$, $g(c_1)$, and since $f(x)-h_1(x)\geq 0$, $c_1\leq x\leq b$, the integral from c_1 to b can be upper bounded by replacing g(x) by its maximum value on $[c_1,b]$, $g(c_1)$. Thus, I can be upper bounded as

$$I \leq \int_a^b g(x)h_1(x)dx + g(c_1)\int_a^{c_1}[f(x)-h_1(x)]dx + g(c_1)\int_{c_1}^{b}[f(x)-h_1(x)]dx$$

or

$$I \leq \int_a^b g(x)h_1(x)dx + g(c_1)\int_a^b [f(x)-h_1(x)]dx. \tag{5}$$

Selecting $h_1(x)$ such that the integral from a to b of $f(x)-h_1(x)$ equals zero yields

$$I \leq \int_a^b g(x)h_1(x)dx. \tag{6}$$

Likewise, since $f(x)-h_2(x)\geq 0$, $a\leq x\leq c_2$, the integral from a to c_2 can be lower bounded by replacing g(x) by its minimum value on $[a,c_2]$, $g(c_2)$, and since $f(x)-h_2(x)\leq 0$, $c_2\leq x\leq b$, the integral from c_2 to b can be lower bounded by replacing g(x) by its maximum value on $[c_2,b]$, $g(c_2)$. Thus, I can be lower bounded as

$$I \geq \int_a^b g(x)h_2(x)dx + g(c_2)\int_a^{c_2}[f(x)-h_2(x)]dx + g(c_2)\int_{c_2}^{b}[f(x)-h_2(x)]dx$$

or

$$I \geq \int_a^b g(x)h_2(x)dx + g(c_2)\int_a^b [f(x)-h_2(x)]dx. \tag{7}$$

Selecting $h_2(x)$ such that the integral from a to b of $f(x)-h_2(x)$ equals zero yields

$$I \geq \int_a^b g(x)h_2(x)dx. \tag{8}$$

Combining (6) and (8) gives the desired bounds

$$\int_a^b g(x)h_2(x)dx \le \int_a^b g(x)f(x)dx = I \le \int_a^b g(x)h_1(x)dx \tag{9}$$

using the alternating functions $f(x)-h_1(x)\le 0$, $a\le x\le c_1$, $f(x)-h_1(x)\ge 0$, $c_1\le x\le b$, and $f(x)-h_2(x)\ge 0$, $a\le x\le c_2$, $f(x)-h_2(x)\le 0$, $c_2\le x\le b$ for $a\le c_1,c_2\le b$.

MPSK WORD ERROR PROBABILITY

As an application consider the M-ary phase shift keying (MPSK) word (or symbol) error probability in Gaussian noise. The word error probability for MPSK in white Gaussian noise can be expressed as [1]

$$P_M = 2p - Q(R) + I \tag{10}$$

where

$$I = \int_R^\infty \frac{2}{\sqrt{2\pi}} e^{-z^2/2}\, Q\left[(z-R)\tan\left(\frac{\pi}{M}\right)\right]dz, \tag{11}$$

$p=Q\left[R \sin\left(\frac{\pi}{M}\right)\right]$, $R=\sqrt{\frac{2E_s}{N_0}}$, E_s is the energy per word, $N_0/2$ is the two-sided power spectral density of the noise, and the Q function is defined as

$$Q(a) = \int_a^\infty \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx. \tag{12}$$

Since the Q function can be upper bounded as [2]

$$Q(x) \le \frac{1}{2} e^{-x^2/2}, \tag{13}$$

an upper bound on the integral of (11) is

$$I \le T = \int_R^\infty \frac{1}{\sqrt{2\pi}} e^{-[z^2+(z-R)^2\tan^2(\pi/M)]/2} dz. \tag{14}$$

Figure 1 shows Q(x) and its upper bound $\frac{1}{2} e^{-x^2/2}$. Completing the square in the exponent (14) gives

$$z^2+(z-R)^2\tan^2\left(\frac{\pi}{M}\right) = \left[1+\tan^2\left(\frac{\pi}{M}\right)\right](z-\mu_z)^2+R\mu_z = \frac{(z-\mu_z)^2}{\sigma_z^2}+R\mu_z \tag{15}$$

where $\mu_z=R\sin^2\left(\frac{\pi}{M}\right)$ and $\sigma_z^2=\cos^2\left(\frac{\pi}{M}\right)$. Thus, I is upper bounded as

$$I \leq T = \sigma_z e^{-R\mu_z/2} Q\left(\frac{R-\mu_z}{\sigma_z}\right) = \cos\left(\frac{\pi}{M}\right) e^{-\left[R\sin\left(\frac{\pi}{M}\right)\right]^2/2} Q\left[R\cos\left(\frac{\pi}{M}\right)\right]. \tag{16}$$

A lower bound can be obtained in terms of T using (9) with $f(x)=Q(x)$ and bounding $Q(x)$ by $Q(x)\geq \frac{d}{2} e^{-x^2/2}$, $0\leq x\leq c_2$ and $Q(x)\leq \frac{d}{2} e^{-x^2/2}$, $c_2\leq x<\infty$, where d is obtained by setting

$$\int_0^{\infty} \left[\frac{d}{2} e^{-x^2/2} - Q(x)\right]dx = 0 \tag{17}$$

Figure 2 shows the crossing nature of $Q(x)$ and $\frac{d}{2} e^{-x^2/2}$. Using

$$\int_0^{\infty} Q(x)dx=\frac{1}{\sqrt{2\pi}}, \tag{18}$$

(17) reduces to

$$\frac{d}{4}\sqrt{2\pi} - \frac{1}{\sqrt{2\pi}} = 0$$

which gives $d=2/\pi$. This value of d then yields $c_2=0.6509752442$. Thus, I can be lower bounded as

$$I \geq dT = \frac{2}{\pi} T \tag{19}$$

where T is given in (16).

An improved upper bound can be obtained using (9) with $f(x)=Q(x)$ and bounding $Q(x)$ by $Q(x)\leq h_1(x)=\frac{1}{2} e^{-\tau x^2/2}$, $0\leq x\leq c_1$ and $Q(x)\geq \frac{1}{2} e^{-\tau x^2/2}$, $c_1\leq x<\infty$, where τ is obtained by setting

$$\int_{0}^{\infty}\left[\frac{1}{2}e^{-\tau x^2/2} - Q(x)\right]dx = 0. \tag{20}$$

Figure 3 shows the crossing nature of Q(x) and $\frac{1}{2}e^{-\tau x^2/2}$. Using (18), (20) reduces to

$$\frac{1}{4}\sqrt{\frac{2\pi}{\tau}} - \frac{1}{\sqrt{2\pi}} = 0$$

which gives $\tau=\pi^2/4$. This value of τ then yields $c_1=0.9000621626$. Now, I can be upper bounded by

$$I \le T' = \int_{R}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-[z^2+\tau(z-R)^2\tan^2(\pi/M)]/2}dz. \tag{21}$$

Completing the square in the exponent of (21) gives

$$z^2+\tau(z-R)^2\tan^2\left(\frac{\pi}{M}\right) = \left[1+\tau\tan^2\left(\frac{\pi}{M}\right)\right](z-\mu_z')^2+R\mu_z' = \frac{(z-\mu_z')^2}{\sigma_z'^2}+R\mu_z' \tag{22}$$

where $\mu_z' = \dfrac{\tau R\tan\left(\frac{\pi}{M}\right)}{1+\tau\tan^2\left(\frac{\pi}{M}\right)}$ and $\sigma_z'^2 = \dfrac{1}{1+\tau\tan^2\left(\frac{\pi}{M}\right)}$. Thus, I is upper bounded by

$$I \le T' = \sigma_z' e^{-R\mu_z'/2}Q\left(\frac{R-\mu_z'}{\sigma_z}\right) = \frac{1}{\sqrt{1+\tau\tan^2\left(\frac{\pi}{M}\right)}}e^{-\frac{\tau R^2\tan^2\left(\frac{\pi}{M}\right)}{2\left[1+\tau\tan^2\left(\frac{\pi}{M}\right)\right]}}Q\left[\frac{R}{\sqrt{1+\tau\tan^2\left(\frac{\pi}{M}\right)}}\right]. \tag{23}$$

Putting (16) and (23) into (10) then yields the MPSK upper and lower bounds

$$2p-Q(R)+\frac{2}{\pi}T \le P_M \le 2p-Q(R)+T' \tag{24}$$

Comparison of the bounds of (24) and more complicated bounds of this type relative to previously developed bounds for MPSK [3-7] are given in [1].

CONCLUSIONS

A procedure for obtaining tighter upper and lower bounds on integrals using alternating functions has been developed. Bounds obtained from this procedure have been developed to upper and lower bound MPSK word error probability.

REFERENCES

[1] J. J. Komo and K. D. Barnett, "Improved Coherent M-ary Phase Shift Keying Bounds," submitted to *IEEE Trans. on Communications*.

[2] J. M. Wozencraft and I. M. Jacobs, Principles of Communication Engineering. Prospect Heights, IL: Waveland Press, 1990.

[3] E. Arthurs and H. Dym, "On the Optimum Detection of Digital Signals in the Presence of White Gaussian Noise - A Geometric Interpretation and a Study of Three Basic Data Transmission Systems," *IRE Trans. on Commun. Syst.*, vol. CS-10, No. 4, pp. 336-372, Dec. 1962.

[4] C. M. Chie, "Bounds and approximations for rapid evaluation of coherent MPSK error probabilities," *IEEE Trans. Commun.*, vol. COM-33, pp. 271-273, Mar. 1985.

[5] V. K. Prabhu, "Error-rate considerations for coherent phase-shift keyed systems with co-channel interference," *Bell Syst. Tech. J.*, vol. 48, pp. 743-767, Mar. 1969.

[6] M. I. Irshid and I. S. Salous, "Bit Error Probability for Coherent M-ary PSK Systems," *IEEE Trans. on Commun.*, vol. COM-39, pp. 349-352, Mar. 1991.

[7] P. L. Lee, "Computation of the bit error rate of M-ary PSK with Gray code bit mapping," *IEEE Trans. Commun.*, vol. COM-34, pp. 488-491, May 1986.

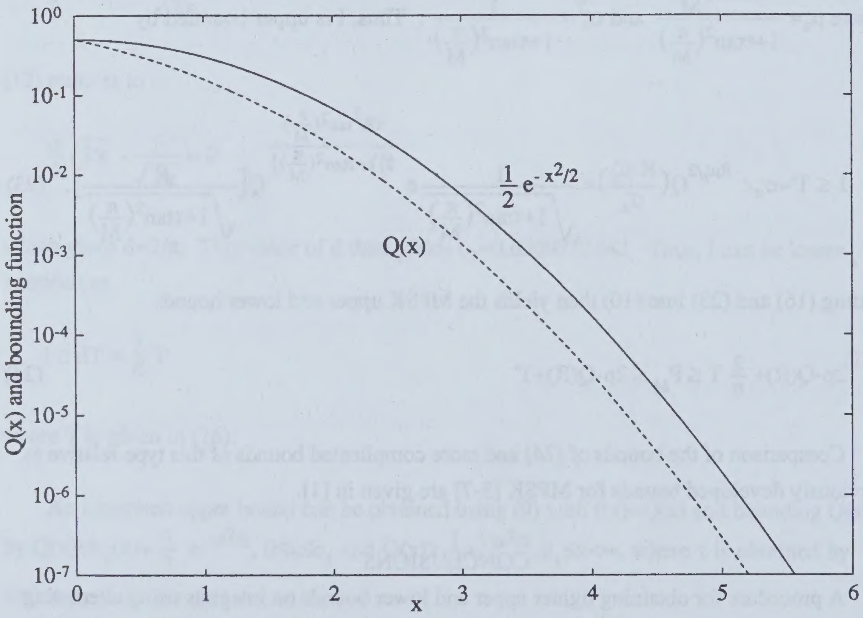


Figure 1. Strict upper bound.

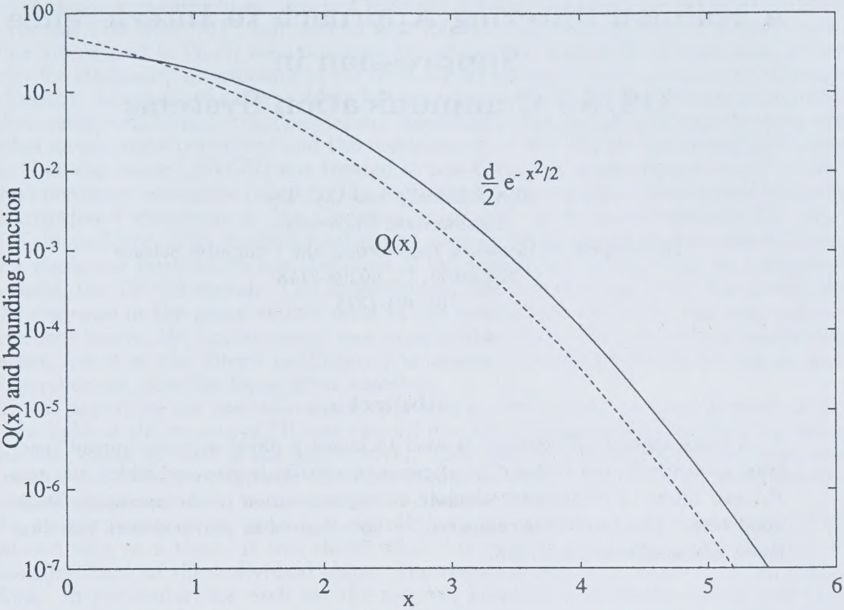


Figure 2. Alternating lower bounding function.

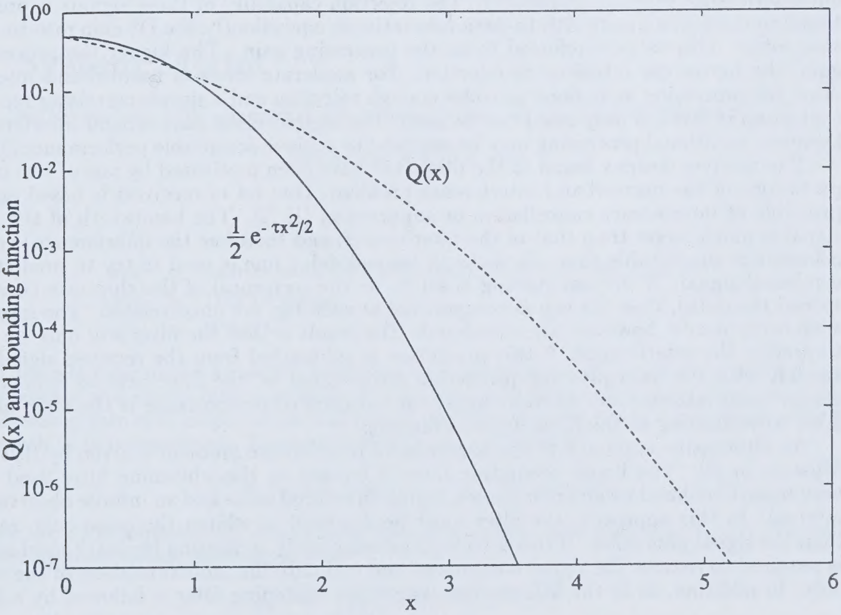


Figure 3. Alternating upper bounding function.

A Kalman Filtering Approach to Interference Suppression in DS/SS Communication Systems

David Krinsky and C.C. Lee
Northwestern University
Department of Electrical Engineering and Computer Science
Evanston, IL, 60208-3118
708-491-7375

Abstract

A state-space representation is used to model a direct-sequence spread spectrum system with narrowband interference. A receiver is proposed which uses two Kalman filters to recursively calculate an approximation to the maximum likelihood ratio. This receiver is compared to those derived in previous work based on linear and nonlinear prediction.

1 Introduction

The primary advantage of direct sequence spread spectrum signals (DS/SS) is their inherent interference rejection capability. The rejection capability of these signals is proportional to the signal bandwidth-to-data rate ratio, or equivalently, the DS chip rate-to-data rate ratio. This ratio is referred to as the processing gain. The higher the processing gain, the better the interference rejection. For moderate levels of narrowband interference, the processing gain alone provides enough rejection and a simple correlator receiver (optimum in AWGN only case) can be used. For high levels of narrowband interference, however, additional processing may be required to achieve acceptable performance [3]-[7].

The receiver designs found in the literature have been motivated by one of two interpretations of the narrowband interference problem. One set of receivers is based on the principle of interference cancellation or suppression [1] [2]. The bandwidth of the data signal is much larger than that of the interference, and therefore the interference is more coherent or predictable than the data. A tapped delay line is used to try to predict the combined signal. If the tap spacing is set to be the reciprocal of the chip rate (used to spread the data), then the signal components at each tap are uncorrelated. The interference components, however, are correlated. The result is that the filter will only be able to predict the interference. If this prediction is subtracted from the received signal, we are left with the data plus the prediction error. That is, the filter acts to remove the narrowband interference. In most work, the measure of performance is the ratio of the SNR with filtering to the SNR without filtering.

An alternative approach to the narrowband interference problem is given by Iltis and Milstein in [3]. The linear prediction filter is treated as the whitening filter used in a maximum likelihood receiver for known signals in colored noise and an infinite observation interval. In this approach, the filter must be designed to whiten the noise only, rather than the signal plus noise. If this is to be done adaptively, a decision feedback mechanism is required to remove the signal component and estimate the autocorrelation of the noise only. In addition, as in the ML receiver where the whitening filter is followed by a filter matched to the whitening filter, Iltis and Milstein followed the linear prediction (LP) filter with its matched filter (MF) and showed that this does indeed improve the performance.

The filters used in [3]-[7] were designed by solving the Wiener-Hopf equations, either directly for stationary interference or via the LMS algorithm in the case of non-stationary interference. Alternatively, the problem has been formulated [8] as a nonlinear estimation problem using a state-space representation. Specifically, the narrowband interference was modeled as the state trajectory and the combination of the DS/SS signal and addititive white Gaussian noise (AWGN) was treated as non-Gaussian measurement noise. A suboptimal nonlinear estimator based on the results in [9] was used to estimate and subtract the narrowband interference. For Gaussian noise, the optimum estimator is the standard Kalman filter. The nonlinear estimator used in [8] incorporates into the Kalman filter a nonlinear term which has the effect of estimating and subtracting, on a chip-by-chip basis, the DS/SS signal. The improvement over the Kalman filter was measured by the decrease in the mean square error of the interference estimate. For very narrow interference bands, the improvement was considerable. Once the interference bandwidth increases, however, the filter's nonlinearity is unable to estimate the DS/SS signal, and the improvement over the linear filter vanishes.

In this paper, we use the state-space approach to develop an improved receiver structure. In light of the results in [3], one can achieve a performance improvement by using a structure based on that of the ML receiver. This receiver uses two Kalman filters, one assuming a bit 0 was received and the other assuming a bit 1 was received, to recursively generate an approximation to the maximum likelihood ratio.

As another observation note that the filter in [8] attempts to estimate the DS/SS signal one chip at a time. It was shown that this process can be interpreted as a soft decision feedback of the individual chips. The received sequence, however, is not really random. In particular, for each bit the receiver knows it will receive one of two chip sequences (assuming binary signaling). We will show that our receiver structure exploits this knowledge, and the systems processing gain, to generate a soft decision on a bit-by-bit, rather than chip-by-chip, basis and that this structure leads to an improved estimate of the interference and subsequently an improved bit error rate.

2 System Model

The DS/SS system with narrowband interference can be modeled as follows [3]:

$$z[k] = y[k] + s[k] + v[k] \tag{1}$$

where

$$\begin{aligned} z[k] &= \text{received signal} \\ y[k] &= \text{narrowband (w.r.t the DS/SS signal) interference} \\ s[k] &= \text{DS/SS signal} \\ v[k] &= \text{AWGN} \end{aligned}$$

This discrete time model arises when the received continuous time signal is passed through an integrate-and-dump filter operating at the chip rate. We assume binary signaling and a processing gain of G chips/bit so that during each bit interval, one of two code sequences of length G is transmitted. These code sequences are denoted,

$$S_1^G = \{s_1[G], s_1[G-1], ..., s_1[1]\} = -S_0^G \tag{2}$$

where the code sequence S_1^G is a psuedorandom sequence with $s_1[k] \in \{+1, -1\}$. We will denote an observation subsequence as

$$Z_0^k \;=\; \{z[k], z[k-1], ..., z[1]\}$$

We assume that the narrowband noise is an AR process described as follows

$$y[k] = a_1 * y[k-1] + a_2 * y[k-2] + \ldots + a_N * y[k-N] + w[k] \tag{3}$$

where N is the system order and $w[k]$ is a white gaussian noise sequence with variance Q. Let the variance of $v[k]$ be denoted as R. We can then write the familiar state-space system description:

$$\begin{aligned} x[k] &= \Phi x[k-1] + G w[k] \qquad (4) \\ z[k] &= H x[k] + s[k] + v[k] \qquad (5) \end{aligned}$$

where

$$x[k] = [y[k] \quad y[k-1] \quad \ldots \quad y[k-N+1]]^T \tag{6}$$

$$\Phi = \begin{pmatrix} a_1 & a_2 & \ldots & a_{N-1} & a_N \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \tag{7}$$

$$G = [1 \quad 0 \quad 0 \quad \ldots \quad 0]^T \tag{8}$$

and

$$H = [1 \quad 0 \quad 0 \quad \ldots \quad 0] \tag{9}$$

3 Receiver Design

We initially consider the detection of one bit. Since the two code sequences are assumed equiprobable and the error costs are equal, the ML and MAP criteria are equivalent and the likelihood ratio is given by

$$\Lambda = \frac{p(S_1^G / Z^G)}{p(S_0^G / Z^G)} \tag{10}$$

We can calculate this ratio recursively. Using Baye's rule, we can show

$$p(S_i^G / Z^k) = \frac{p(z[k]/S_i^G \, Z^{k-1}) p(S_i^G / Z^{k-1})}{p(z[k]/Z^{k-1})} \tag{11}$$

Assume that the second term in the numerator is available from the previous recursion. The likelihood ratio can then be written

$$\Lambda = \prod_{k=1}^{G} \frac{p(z[k]/S_1^G \, Z^{k-1})}{p(z[k]/S_0^G \, Z^{k-1})} \tag{12}$$

To derive expressions for $p(z[k]/S_1^G \, Z^{k-1})$ and $p(z[k]/S_0^G \, Z^{k-1})$ we run two Kalman filters. One assumes that sequence S_1^G was received, the other assumes S_0^G was received. These filters generate the mean and covariance of $x[k]$ conditioned on $z[k]$ and S_i^G.

$$\begin{aligned} \hat{x}_1[k] &= \mathrm{E}\{x[k]/Z^k \, S_1^G\} \qquad (13) \\ \hat{x}_0[k] &= \mathrm{E}\{x[k]/Z^k \, S_0^G\} \qquad (14) \\ P[k] &= P_1[k] = P_0[k] \\ &= \mathrm{E}\{(x[k] - \hat{x}_i[k])(x[k] - \hat{x}_i[k])^T / Z^k\} \qquad (15) \end{aligned}$$

The two covariance matrices are the same since both filters assume that the $s[k]$ component has been removed (the filter which made the wrong assumption will be biased). We can now show that

$$p(z[k]/S_i^G\,Z^{k-1}) \sim N(\hat{y}_i[k^-] + s_i[k], P_y[k^-] + R) \tag{16}$$

where

$$\begin{aligned} \hat{y}_i[k^-] &= H\Phi\hat{x}_i[k-1] \\ &= \mathrm{E}\{y[k]/Z^{k-1}\,S_i^G\} \end{aligned} \tag{17}$$

and

$$\begin{aligned} P_y[k^-] &= H(\Phi P[k-1]\Phi^T + GQG^T)H^T \\ &= \mathrm{Var}((y[k] - y_i[k^-])/Z^{k-1}\,S_i^G) \end{aligned} \tag{18}$$

and the resulting log likelihood ratio is

$$\begin{aligned} \log\Lambda &= \sum_{k=1}^{G} \frac{1}{P_y[k^-] + R}\{(\hat{y}_1[k^-] + s_1[k])z[k] - (\hat{y}_0[k^-] + s_0[k])z[k] \\ &- \frac{1}{2}((\hat{y}_1[k^-] + s_1[k])^2 - (\hat{y}_0[k^-] + s_0[k])^2)\} \end{aligned} \tag{19}$$

From Equation (19) we can see that the receiver is in the form of an estimator-correlator. This structure is identical to the structure used for detecting known signals in white noise. Here, however, the narrowband interference is treated as an unknown component of the data signals, and the known signals are replaced by their estimates.

Before analyzing the bit error performance of the receiver, we consider its performance as an estimator of the narrowband interference. The receiver generates two estimates of the narrowband interference, $\{\hat{y}_1[1]...\hat{y}_1[G]\}$ and $\{\hat{y}_0[1]...\hat{y}_0[G]\}$. From the likelihood ratio we also have the conditional probability of each sequence. These probabilities are used to weight and combine the two estimates. The resulting sequence, $\{\hat{y}_{\mathrm{cmb}}[1]...\hat{y}_{\mathrm{cmb}}[G]\}$, is actually the MMSE estimate conditioned on Z^G [10]. In effect, we are using a soft decision feedback mechanism to estimate the interference. While the filter in [8] used feedback on a chip-by-chip basis, our filter takes advantage of the knowledge of the two code sequences to calculate a soft decision over the duration of a bit. Simulations have shown that for extremely narrowband interference, the performance of both filters is almost identical - both almost perfectly estimate the signal component $s[k]$. As the bandwidth of the noise increases, however, the filter described in [8] can no longer accurately estimate the signal $s[k]$ and the filter performance is approximately that of the linear filter.

This receiver, when used to detect 1 bit, uses the output of two Kalman filters to generate the MMSE estimate of the interference conditioned on observations Z^G. The two filters correspond to the two possible signal sequences, S_1^G and S_0^G. If we wanted the MMSE estimate of the interference after two bit intervals, we would need four Kalman filters, one for each of the four possible signal sequences. In general, the MMSE estimate over a sequence of bits requires an exponentially increasing number of filters. This is the standard problem found in hybrid system estimation [10] [11].

To use this receiver for a stream of bits, we use the following method. At the end of each bit interval, the two estimates $\{\hat{y}_1[1]...\hat{y}_1[G]\}$ and $\{\hat{y}_0[1]...\hat{y}_0[G]\}$ are combined to form $\{\hat{y}_{\mathrm{cmb}}[1]...\hat{y}_{\mathrm{cmb}}[G]\}$. This combined estimate is then used for the initial conditions (prior estimates) in the detection of the next bit. For low bit error rates, the interference estimate is close to optimal. This estimate is a special case of the suboptimal estimate described in [10].

4 Analysis of a First Order System

We will now analyze the receiver performance for the case of a first order system and show that as the processing gain increases, it becomes equivalent to the ML based receiver described in [3].

For the first order system $x[k] = y[k]$ and

$$\begin{aligned} y[k] &= a * y[k-1] + w[k] && (20) \\ z[k] &= y[k] + s[k] + v[k] && (21) \end{aligned}$$

Due to the symmetry of the problem, we can assume that S_1^G is received so that $s[k] = s_1[k]$.

To digress for a moment, suppose $s[k]$ is known and that our goal is to estimate $y[k]$. We could then form $z'[k] = z[k] - s[k]$ and use the standard Kalman filter equations to generate estimates $\hat{y}[k^-]$ and $\hat{y}[k]$ and error covariance $P[k]$. The filter equations are as follows.

$$\hat{y}[k] = \hat{y}[k^-] + K[k](z[k] - \hat{y}[k^-]) \tag{22}$$

$$K[k] = \frac{P[k^-]}{(P[k^-] + R)} \tag{23}$$

$$P[k] = (1 - K[k])P[k^-] \tag{24}$$

$$P[k^-] = a^2 P[k-1] + Q \tag{25}$$

$$\hat{y}[k^-] = a\hat{y}[k-1] \tag{26}$$

$\hat{y}[k^-]$ is referred to as the predicted estimate and $\hat{y}[k]$ is referred to as the filtered estimate.

Returning to our receiver, note that the log likelihood ratio derived above requires the conditional estimates $\hat{y}_1[k^-]$ and $\hat{y}_0[k^-]$. The receiver uses two Kalman filters, both of which assume that signal $s[k]$ is known. Only one of these filters is correct. The first filter assumes, correctly in this case, that S_1^G is received, properly removes $s[k]$ to produce $z_1[k] = y[k] + v[k]$, and generates $\hat{y}_1[k] = \hat{y}[k]$, the optimal estimate of $y[k]$.

The second filter assumes, incorrectly, that S_0^G is received. Since $s_1[k] = -s_0[k]$, this filter actually doubles the signal term. The subtraction in this case produces $z_0[k] = y[k] + 2s_1[k] + v[k]$.

Since both filters assume that $s[k]$ has been properly removed, the two filters are actually identical but operate on different observations. As a result, the gain terms $K[k]$ and error terms $P[k^-]$ and $P[k]$ are the same for both filters. As mentioned above, filtering $z_1[k] = y[k] + v[k]$ generates the optimal estimate $\hat{y}[k]$. When $z_0[k] = y[k] + 2s_1[k] + v[k]$ is filtered, the output is $\hat{y}_0[k] = \hat{y}[k] + y_s[k]$ where $y_s[k]$ is a bias due to the doubled signal term. It can be shown that

$$y_s[k] = (1 - K[k])a * y_s[k-1] + 2K[k]s_1[k] \tag{27}$$

where $K[k]$ is the gain term from the Kalman filter. Note that the bias term $y_s[k]$ does not depend on the data. It can be generated off-line by running the sequence S_1^G through a filter with transfer function $2K[k]/(1 - (1 - K[k])az^{-1})$.

By substituting the above results into the likelihood ratio and rearranging terms, we can get the following expression

$$\begin{aligned} \log \Lambda &= \sum_{k=1}^{G} \frac{1}{P[k^-] + R}\{(2s_1[k] - y_s[k^-])[(y[k] - \hat{y}[k^-]) + v[k]] \\ &\quad + 2(s_1[k] - y_s[k^-])s_1[k] + \frac{1}{2}y_s^2[k^-]\} && (28) \end{aligned}$$

This statistics of this expression can easily be found and used to calculate the probability of error.

We will now show the relationship between this filter and the ML based receiver in [3]. To simplify the comparison, we will assume that there is no measurement noise, i.e. $v[k] = 0$. The purpose in this is to simplify the linear prediction filter. With $v[k] = 0$, a simple one-tap linear prediction filter perfectly whitens the signal, while in general, the perfect linear prediction filter requires an infinite number of taps.

Recall that the receiver in [3] is based on the ML receiver for colored noise and an infinite observation interval. The ML receiver consists of a whitening filter in cascade with its matched filter, the output of which is correlated with the signal waveform $s[k]$. In this example, the resulting decision statistic would be as follows

$$D1 = \sum_{k=1}^{G} (z[k] - \frac{1}{1+a^2} z[k-1] - \frac{1}{1+a^2} z[k+1]) s[k] \tag{29}$$

An equivalent structure for the ML receiver correlates the whitened received signal with the whitened signal waveform. For this example, this would lead to the following decision statistic

$$\begin{aligned} D2 &= \sum_{k=1}^{G} (z[k] - az[k-1])(s[k] - as[k-1]) \\ &= \sum_{k=1}^{G} (y[k] - ay[k-1])(s[k] - as[k-1]) \\ &\quad + \sum_{k=1}^{G} (s[k] - as[k-1])^2 \end{aligned} \tag{30}$$

These two ML structures are only equivalent when used with an infinite observation interval. In practice, we use an observation interval of one bit.

Comparing Equations (29),(30) and (28), we make the following observations. The difference between the two ML structures described by Equations (29) and (30) is due to the effect of the finite observation interval. If the bit interval is reasonably long, then this differerence is neglible. Similarly, the difference between Equations (28) and (30) is the weighting factors of $P[k^-] + R$, which converge to a constant steady state value. When the bit interval is long compared to the convergence time, the difference between Equations (28) and (30) is neglible. The conclusion is that for moderately large processing gains, the state-space receiver described in this paper is equivalent to the LP/MF receiver described in [3].

5 Simulation Results

Since our receiver is based on the ML receiver, we will refer to it as the Approximate Maximum Likelihood (AML) receiver. A computer simulation was used to measure the performance of the AML receiver and compare this performance to that of the ACM filter/receiver. Both the SNR improvement and the bit error rate (BER) have been evaluated. As in [8], the SNR improvement is defined as follows:

$$\text{SNR improvement} = \frac{\mathrm{E}\{(z[k] - s[k])^2\}}{\mathrm{E}\{(e[k] - s[k])^2\}} \tag{31}$$

where

$$e[k] = z[k] - \hat{y}[k] \tag{32}$$

or

$$e[k] = z[k] - \hat{y}[k^-] \tag{33}$$

for the filtered SNR improvement and the predicted SNR improvement respectively. For our simulation, we considered a first-order system with state equation

$$x[k] = 0.9x[k-1] + w[k] \tag{34}$$

The interference power, $E\{x^2[k]\}$, is varied from 10.0 to 1000.0. Simulations are run using a measurement noise power R of 0.01 and 1.00. The results are shown in Figure 1. Figure 1.A compares the filters' SNR improvement, using both the predicted estimate and the filtered estimate, for a measurement noise power of 0.01. Figure 1.B shows the same comparison for a measurement noise power of 1.00. In both cases the AML receiver outperforms the ACM filter/receiver. The explanation is as follows. The SNR improvement indicates how well the narrowband interference is estimated. When trying to estimate this interference, the DS/SS data signal acts as measurement noise. Both receivers try to estimate and remove the data signal and thus decrease the measurement noise. The ACM filter does this while treating the data signal as an independent binary sequence. The DS/SS data signal, however, is not random. During each bit interval, it is one of two known sequences. The AML receiver uses this knowledge to improve its estimate of the data signal, thereby reducing the measurement noise and improving the estimate of the narrowband interference. More specifically, as the likelihood ratio is updated at each iteration, it provides the probability of each of the two sequences and thus an approximately optimal estimate of the data signal (the approximation is due to the estimate combining done following each bit). This improved estimate of the data signal leads to an improved estimate of the narrowband interference.

The most important measure of performance for a communication system is the bit error rate. Figures 1.C and 1.D show the BER of the two filters for a measurement noise power of 0.01 and 1.00 respectively. Again the AML receiver outperforms the ACM filter/receiver. The performance advantage of the AML receiver can be explained by considering the distortion of the data signal due to filtering. The ACM filter ignores this distortion, resulting in a receiver which is no longer matched to the data signal. The AML filter, however, uses a second Kalman filter to effectively match the receiver to the filtered data signal. This is basically the result described in [3] only here the implementation is based on the state-space system representation.

Lastly, note that the SNR improvement of the systems converge as the narrowband interference power increases. The reason is that the effective measurement noise (the data signal plus the AWGN measurement noise) becomes negligible compared to the narrowband interference.

6 Conclusions

In this paper we have used a state-space system representation to derive an approximate Maximum Likelihood receiver for DS/SS signals in narrowband interference. The performance of this receiver was evaluated through computer simulations and compared to that of the ACM filter/receiver. The results show that by using the knowledge of the DS/SS code sequence, the AML receiver offers improved performance as measured by both the SNR improvement and BER.

It was also shown that for time-invariant systems the AML receiver is basically the state-space equivalent of the Wiener filter based receiver described in [3]. However, the narrowband noise may not be stationary and using this Kalman filtering approach can offer some advantages. For example, when using an LMS algorithm to track the narrowband interference, we would like to remove the data signal from the observations [3] [5] [8]. It was shown that the AML receiver provides an approximately optimal estimate of the data signal.

Finally, there are some other system models which are time-varying that can be considered. Examples include impulsive noise and/or pulsed narrowband interference. Since state-space filtering can more easily handle time-varying systems, the receiver described in this paper could be modified for use in such systems.

References

[1] B. Widrow, J.R. Glover Jr., J.M. McCool, J. Kaunitz, C.S. Williams, R.H. Hearn, J.R. Zeidler, E. Dong Jr., and R.C. Goodlin, "Adaptive noise cancelling: principles and applications," *Proc. IEEE,* vol. 63, no. 12, December 1975, pp. 1692-1716.

[2] F.M. Hsu and A.A. Giordano, "Digital whitening techniques for improving spread spectrum communications performance in the presence of narrowband jamming and interference," *IEEE Trans. Comm.* , vol. COM-26, no. 2, February 1978, pp. 209-216.

[3] R. Iltis and L. B. Milstein, " Performance analysis of narrow-band interference rejection techniques in DS spread-spectrum systems," *IEEE Trans. Comm.,* vol. COM-32, no. 11, pp. 1169-1177, November 1984.

[4] L.M. Li and L. B. Milstein, " Rejection of narrow-band interference in PN spread-spectrum systems using transversal filters,"*IEEE Trans. Comm.,* vol. COM-30, no. 5, pp. 925-928, May 1982.

[5] L.M. Li and L. B. Milstein, " Rejection of Pulsed CW interference in PN spread-spectrum systems using complex adaptive filters," *IEEE Trans. Comm.,* vol. COM-31, no. 1, pp. 10-20, January 1983.

[6] L. B. Milstein, "Interference rejection techniques in spread spectrum communications,"*Proc. IEEE,* vol. 76, no. 6, June 1988, pp. 657-671.

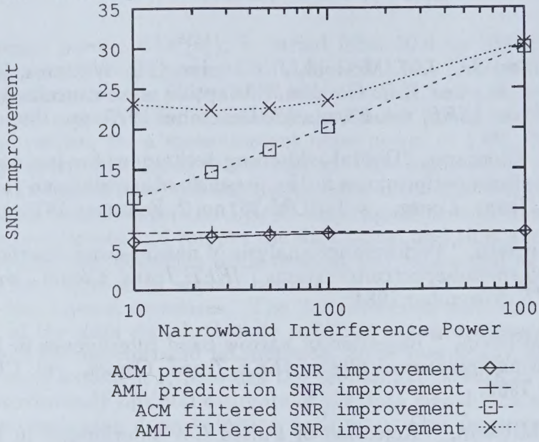
[7] R. L. Pickholtz, L. B. Milstein, and D. L. Schilling, "Spread spectrum for mobile communications," *IEEE Trans. Veh. Tech.,* vol. 40, no. 2, pp. 313-321, May 1991.

[8] R. Vijayan and H.V. Poor, "Nonlinear techniques for interference suppression in spread-spectrum systems," *IEEE Trans. Comm.,* vol. 38 , no. 7 ,pp. 1060-1065, July 1990.

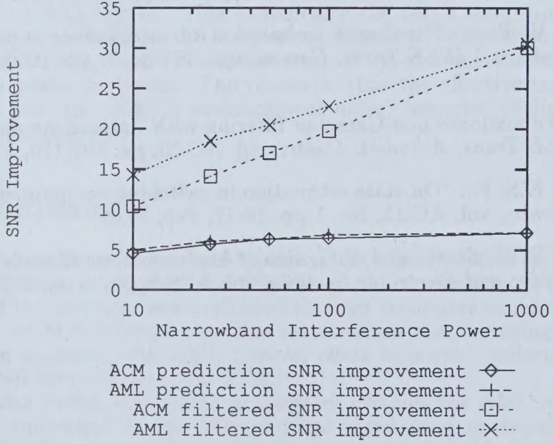
[9] H.W. Masreliez, "Appoximate non-Gaussian filtering with linear state and observation relations," *IEEE Trans. Automat. Contr.,* vol. AC-20, pp. 107-110, Feb. 1975.

[10] G.A Ackerson and K.S. Fu, "On state estimation in switching environments," *IEEE Trans. Automat. Contr.,* vol. AC-15, No. 1 pp. 10-17, Feb. 1970.

[11] J.M.H. Bruckner, R.W. Scott, and F.G. Rea, "Analysis of multimode systems," *IEEE Trans. Aerospace and Electronic Systems,* vol. AES-9, No. 6 pp. 883-888, Nov. 1973.

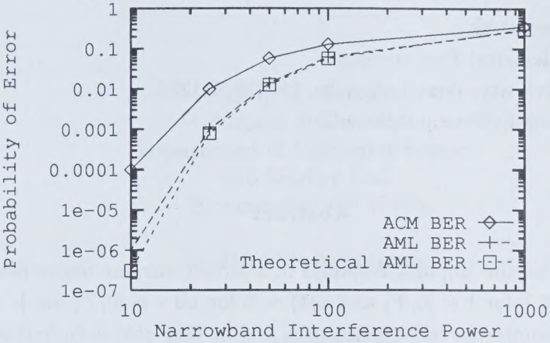


1.A. SNR Improvement vs. NB Interference power. R=0.01

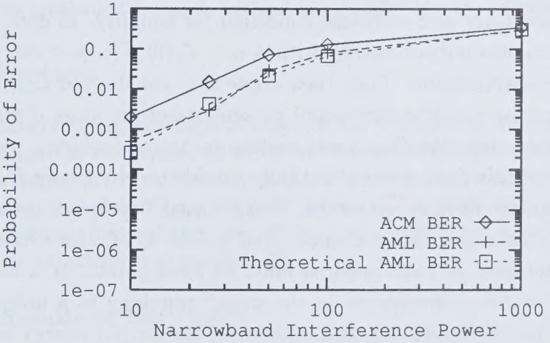


1.B. SNR Improvement vs. NB Interference power. R=1.00

1.C. BER vs. NB Interference power. R=0.01



1.D. BER vs. NB Interference power. R=1.00



On Maximizing Linear System Output Energy with a Time- and Amplitude-Limited Input[1]

John S. Sadowsky
School of Electrical Engineering
Purdue University, West Lafayette, IN 47907-1285
e-mail: sadowsky@ecn.purdue.edu

Abstract

Let $h \in \mathcal{L}_2(\mathbf{R})$ be the impulse response of a time-invariant linear filter, and define $\mathcal{U} = \{u(t)\colon\ |u(t)| \leq 1 \text{ for } t \in [0,T] \text{ and } u(t) = 0 \text{ for all } t \notin [0,T] \text{ a.e.}\}$. Let $h*u(t) = \int_0^T h(t-s)u(s)\,ds$ denote the convolution of h and u. Our goal is to find an input signal $u \in \mathcal{U}$ that maximizes the output energy $\mathcal{E}(u) = \int_{-\infty}^{\infty} |h*u(t)|^2\,dt$.

Define the filter autocorrelation $R_h(\tau) = \int_{-\infty}^{\infty} h(t+\tau)h(t)\,dt$ and the nonlinear transformation $\mathcal{F}u(t) = \text{sgn}(R_h*u(t))$ for $t \in [0,T]$ and $\mathcal{F}u(t) = 0$ for $t \notin [0,T]$. ($\text{sgn}(x) = -1/0/1$ for $x </=/> 0$.) Honig & Steiglitz (*IEEE Trans. Inform. Theory*, pp. 1041-1052, May 1992) have establish that the fixed point condition $\hat{u} = \mathcal{F}\hat{u}$ is a necessary condition for $\mathcal{E}(\hat{u}) = \sup_{u\in\mathcal{U}} \mathcal{E}(u)$, and, in some numerical examples, it was shown that the iteration $u_{n+1} = \mathcal{F}u_n$ can be a good way to find these fixed points.

This correspondence presents a new proof of the necessity of the fixed point condition. At the core of this new proof is an application of the $p = \infty$ and $q = 1$ version of Hölder's inequality and its necessary and sufficient condition for equality. In comparison, Honig & Steiglitz first maximize output energy with a $u \in \mathcal{L}_p([0,T])$, $p < \infty$, constraint via Lagrange multiplier optimization. They then argue that the desired $\mathcal{L}_\infty([0,T])$ result is obtained by letting $p \to \infty$. The new proof presented here is more direct; there is no $\mathcal{L}_p([0,T])$ intermediate step. We thus avoid certain technical problems with the $p \to \infty$ limit. (The validity of this limit was not actually established by Honig & Steiglitz.) We also improve the conclusions in several ways. Using a weak* compactness and continuity argument, we easily establish the existence fixed points in $\mathcal{U}$, the existence of global maxima, and the fact that all such maxima must be fixed points. It is also shown that the iterations $u_{n+1} = \mathcal{F}u_n$ convergence in the weak* topology to a unique fixed point (determined by the initial iterate u_0), and, moreover, the output energy sequence $\mathcal{E}(u_n)$ is strictly increasing sequence up to convergence.

The fixed points are signals $u \in \mathcal{U}$ such that $u(t) = \pm 1$ a.e. in $[0,T]$, and hence, such signals are determined by their zero crossings. It was conjectured that there might be at most countably many fixed points, however, we give a simple example where there is actually a continuum of fixed points. We also show that by imposing smoothness conditions on the impulse response $h(\cdot)$, for example, if $h(\cdot)$ is bandlimited, then all fixed points will have at most finitely many zero crossings.

---

[1] This work was supported by the National Science Foundation, grant No. 9003007-NCR.

Efficient Parallel Dynamic Programming

Phillip G. Bradford
Indiana University
Department of Computer Science
215 Lindley Hall
Bloomington, IN 47405

bradford@cs.indiana.edu

Abstract

In 1983, Valiant, Skyum, Berkowitz and Rackoff showed that many problems with simple $O(n^3)$ sequential dynamic programming solutions are in the class $\mathcal{NC}$. They used straight line programs to solve these problems using $O(\log^2 n)$[1] time and n^9 processors. In 1988, Rytter used pebbling games to show that these same problems can be solved in $O(\log^2 n)$ time with $n^6/\log n$ processors.

Using a graph structure that is analogous to the classical dynamic programming table, this paper gives an algorithm that improves these results. First, this graph characterization leads to a polylog time and $n^6/\log n$ processor algorithm that solves these problems. Second, there follows a subpolylog time and sublinear processor parallel approximation algorithm for the matrix chain ordering problem. Finally, this paper presents a $n^3/\log n$ processor and $O(\log^3 n)$ time algorithm that solves the optimal matrix chain ordering problem and the problem of finding an optimal triangulation of a convex polygon.

1 Introduction

The dynamic programming paradigm is based on the *principle of optimality.* This principle is that for a structure to be optimal all of its well-formed substructures must also be optimal. Conversely, the *greedy principle* basically is that if a substructure is optimal then it is in some optimal superstructure. Intuitively, the main results of the paper rely on finding applications of the greedy principle that enhance the efficiency of applying the principle of optimality.

This paper is an extended abstract of [2].

1.1 Main Results of this Paper

The *matrix chain ordering* problem (MCOP): Find an optimal ordering to multiply a matrix chain together, where the matrices are pairwise compatible but of varying dimensions. These algorithms also find an optimal triangulation of a convex polygon, see [4].

This paper shows how to transform the MCOP to a minimum cost parenthesization problem on a weighted semigroupoid then to a shortest path problem on a special weighted digraph. Next applying a shortest path algorithm based on matrix multiplication gives a polylog time and $n^6/\log n$ processor solution. Further exploiting special properties of the digraphs modeling the semigroupoids gives a polylog time algorithm using $n^3/\log n$ processors. We also report a subpolylog and sublinear processor approximation algorithm for the MCOP.

---

[1] **This paper takes all logarithms as base 2 and assumes the Common CRCW PRAM parallel model.**

1.2 Previous Results

Using straight line arithmetic programs, Valiant et al. [13] showed that many classical optimization problems with efficient sequential dynamic programming solutions are $\mathcal{NC}$. However, their algorithms appear to require $\Theta(\log^2 n)$ time and n^9 processors. Using pebbling games, Rytter [12] describes a general method to generate more efficient parallel algorithms for a similar class of optimization problems in $O(\log^2 n)$ time with $n^6/\log n$ processors.

The matrix chain ordering problem can be solved sequentially in $O(n^3)$ time with elementary dynamic programming algorithms. However, the best serial solution of the MCOP is Hu and Shing's $O(n \log n)$ algorithm [6, 8, 9]. It has been conjectured that Hu and Shing's algorithm is optimal.

1.3 Structure of this Paper

Section 2 defines weighted semigroupoids; these structures model problems such as the MCOP. In turn weighted semigroupoids are modeled by special *non*planar weighted digraphs. Finding a shortest path in these graphs minimizes an associative product of a weighted semigroupoid. In section 3 a list of weights represents the dimensions of the matrices and combinations of these weights make up the edge weights of the corresponding graph. We model these weights and related graph nodes by the nesting levels of parentheses using the all nearest smaller value problem of Berkman et al. [1]. Section 4 gives an approximation algorithm. Section 5 gives a polylog time parallel algorithm for the MCOP using $n^3/\log n$ processors. This algorithm works by isolating subgraphs and then connecting them to form a tree. Finally, a tree contraction algorithm computes a shortest path.

2 Definitions

This section contains definitions of structures that model associative products.

A (finite) *semigroupoid* $(S, R, \bullet)$ is a nonempty finite set S, a binary relation $R \subseteq S \times S$, and an associative binary operator "$\bullet$" on R satisfying the following conditions [11]:

1. If $(a,b) \in R$, then $a \bullet b \in S$.
2. $(a \bullet b, c) \in R$ iff $(a, b \bullet c) \in R$ and $(a \bullet b) \bullet c = a \bullet (b \bullet c)$.
3. If $(a,b), (b,c) \in R$, then $(a \bullet b, c) \in R$.

An *associative product* is any product of the form $a_1 \bullet a_2 \bullet \cdots \bullet a_n$ such that $(a_i, a_{i+1}) \in R$, for $1 \leq i < n$. A *linear* product is a product of the form $((\cdots(a_1 \bullet a_2) \bullet \cdots) \bullet a_n)$ or $(a_1 \bullet (\cdots \bullet (a_{n-1} \bullet a_n))\cdots)$.

A *weighted* semigroupoid $(S, R, \bullet, pc)$ is a semigroupoid $(S, R, \bullet)$ with a non-negative product cost function pc such that if $(a_i, a_k) \in R$ then $pc(a_i, a_k)$ is the cost of evaluating $a_i \bullet a_k$. The minimal cost of evaluating an associative product $a_i \bullet a_{i+1} \bullet \cdots \bullet a_k$ is $mp(i,k)$. That is,

$$mp(i,k) = \min_{i \leq j < k} \{\ mp(i,j) + mp(j+1,k) + f(i,j,k)\ \}$$

where $f(i,j,k) = pc(a_i \bullet a_{i+1} \bullet \cdots \bullet a_j,\ a_{j+1} \bullet a_{j+2} \bullet \cdots \bullet a_k)$.

Given a weighted semigroupoid and an associative product $a_1 \bullet a_2 \bullet \cdots \bullet a_n$ the problem of finding $mp(1,n)$ solves the MCOP and other problems.

Assume a globally accessible 4-tuple $(S, R, \bullet, pc)$ represents a weighted semigroupoid. Represent pc, R, and $\bullet$ by the array $f(1..n, 1..n, 1..n)$ where

$$f(i,j,k) = \begin{cases} pc(a_i \bullet \cdots \bullet a_j,\ a_{j+1} \bullet \cdots \bullet a_k) & \text{if } (a_i \bullet \cdots \bullet a_j) \bullet (a_{j+1} \bullet \cdots \bullet a_k) \in S \\ \infty & \text{otherwise} \end{cases}$$

Given a_i and a_j, assume that both $a_i \bullet a_j$ and $pc(a_i, a_j)$ are computable in constant time. Also assume the standard elementary matrix multiplication algorithm where the cost of multiplying a $w_i \times w_j$ matrix by a $w_j \times w_k$ matrix is $w_i w_j w_k$ operations, thus $f(i,j,k) = w_i w_j w_k$.

Next we present graph structures on which we can solve the MCOP. An elementary solution of an instance of the MCOP with n matrices needs a graph with $\Theta(n^2)$ nodes. Although, as we will see, we can solve such an instance of the MCOP with only a linear number of nodes in a very restrictive form. The jumper $(i,j) \Longrightarrow (i,k)$ represents the product $(a_i \bullet \cdots \bullet a_j) \bullet (a_{j+1} \bullet \cdots \bullet a_k)$ and weighs $mp(j+1,k) + f(i,j,k)$. When finding a shortest path to (i,k) we take it on faith that a shortest path to $(j+1,k)$ will be computed in time for $mp(j+1,k)$ to be added to $f(i,j,k)$. Similar observations hold for vertical jumpers.

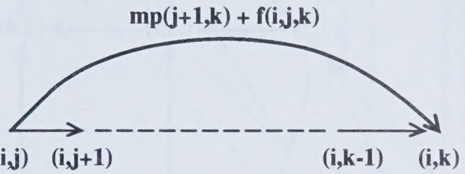


Figure 1: The Horizontal Jumper $(i,j) \longrightarrow (i,k)$ with its Associated Weight

Definition 1 *The weighted digraph $D_n = (V, E \cup E')$, has vertices,*

$$V = \{(i,j) : 1 \le i \le j \le n\} \cup \{(0,0)\}$$

and unit *edges,*

$$\begin{aligned} E \;=\; & \{(i,j) \rightarrow (i,j+1) : 1 \le i \le j < n\} \cup \\ & \{(i,j) \uparrow (i-1,j) : 1 < i \le j \le n\} \cup \\ & \{(0,0) \nearrow (i,i) : 1 \le i \le n\} \end{aligned}$$

together with the jumpers,

$$\begin{aligned} E' \;=\; & \{(i,j) \Longrightarrow (i,t) : 1 < i < j < t \le n\} \cup \\ & \{(s,t) \Uparrow (i,t) : 1 \le i < s < t \le n\} \end{aligned}$$

and each edge has weight,

$$\begin{aligned} W((i,j) \Longrightarrow (i,t)) &= mp(j+1,t) + f(i,j,t), & 1 < i < j < t \le n \\ W((s,t) \Uparrow (i,t)) &= mp(i,s-1) + f(i,s-1,t), & 1 \le i < s < t \le n \end{aligned}$$

in addition let $mp(i,i) = 0$. For example see the graph D_4 in Figure 2.

Note the similarity of a D_n graph and a classical dynamic programming table, T, for the matrix chain ordering problem. The value of $mp(i,k)$ in a D_n graph is the same as $T[i,k]$ in the equivalent dynamic programming table. Calculating a shortest path to (i,k) gives the minimum cost parenthesization of $a_i \bullet \cdots \bullet a_k$. Therefore finding a shortest path from (i,k) to $(1,n)$ gives a minimal parenthesization of $a_1 \bullet \cdots \bullet a_{i-1} \bullet (a_i \bullet \cdots \bullet a_k) \bullet a_{k+1} \bullet \cdots \bullet a_n$.

Theorem 1 *Finding a shortest path from $(0,0)$ to $(1,n)$ in a D_n graph solves the minimal cost parenthesization problem for an associative product with n elements.*

Constructing a D_n graph can be done by starting with a digraph without jumpers and then performing incremental *path relaxation* [4] while adding new jumpers by using a modified $(\min, +)$ matrix multiplication based all pairs minimum path algorithm. This costs $O(\log^2 n)$ time with $n^6/\log n$ processors and solves an n matrix instance of the MCOP and other problems.

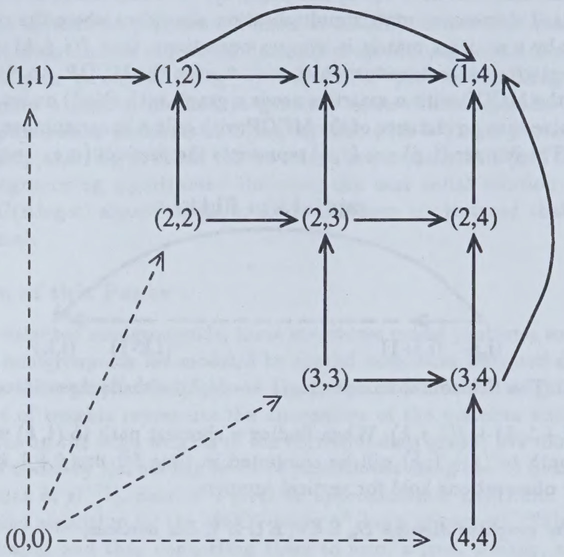


Figure 2: The Weighted Graph D_4

3 Matrix Dimensions as Nesting Levels of Matching Parentheses

This section discusses a technique that allows matrix dimensions to approximate the nesting level of matched parentheses.

Given an associative product where the level of each parenthesis is known, then for each parenthesis we can find its matching parenthesis by solving the all nearest smaller value (ANSV) problem [1]: Given $w_1, w_2, \ldots, w_n$ drawn from a totally ordered set, for each w_i find the largest j, where $1 \leq j < i$, and smallest k where $i < k \leq n$, so that $w_j < w_i$ and $w_k < w_i$ if such values exist. The next definition is originally due to Hu and Shing [6], although they present it in a different framework: In D_n a *critical node* is a node (i, k) such that $w_j > \max\{w_i, w_{k+1}\}$ for all $j, i < j < k + 1$ and call the pair $[w_i, w_{k+1}]$ an ANSV *match*.

By solving the ANSV problem we can compute all critical nodes of a D_n graph. In Figure 3 the weight list is at a 45° angle and four key ANSV matches are represented by dotted lines. The four corresponding critical nodes are circled in the related D_n graph.

The next theorem was proved for ANSV matches by Berkman et al. [1] and here it follows from the relationship of matches in the weight list and critical nodes in D_n graphs.

Theorem 2 *All critical nodes can be computed in $O(\log\log n)$ time using $n/\log\log n$ processors.*

Critical nodes (i, s) and (j, t) are not *compatible*, when $s - i = t - j$ and there exists some vertex other than $(0, 0)$ that can reach both (i, s) and (j, t) by a unit path. In any instance of the matrix chain ordering problem all critical nodes are compatible [8]. Also a D_n graph can have at most $n - 1$ critical nodes but there is always at least one path from $(0, 0)$ to $(1, n)$ that includes all critical nodes.

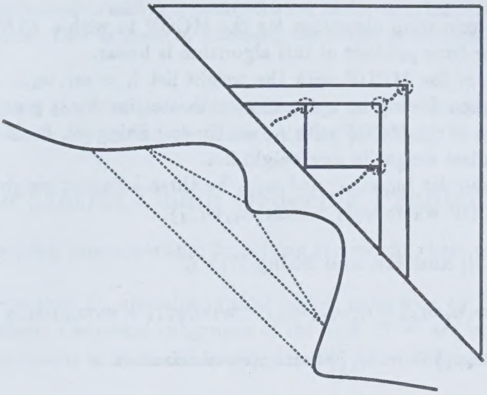


Figure 3: Several Canonical Subgraphs and Their Weight List

3.1 Canonical Subgraphs of D_n

This subsection investigates the interaction between monotonicity and critical nodes.

A subgraph $D(i,t)$ of D_n is all vertices and edges of D_n that can reach (i,t) by a unit path. Formally, $D(i,t)$ is all vertices $(j,k) \in V[D_n]$ such that $i \le j \le k \le t$ in addition to the associated edges. Also, when a graph $D(i,j)$ is such that $w_i, \ldots, w_{j+1}$ is monotonic then the *graph* $D(i,j)$ is monotonic.

A *canonical subgraph* $D^{(i,t)}_{(j,k)}$ is the subgraph containing the maximal contiguous unit edge-connected path of critical nodes that begins at critical node (j,k) and terminates at critical node (i,t) and has vertex set $\{V[D(i,t)] - V[D(j+1,k-1)]\} \cup \{(0,0)\}$ and associated edges. We write $D^{(i,t)}$ for a canonical subgraph of the form $D^{(i,t)}_{(j,j+1)}$, and p denotes the path of critical nodes in a canonical subgraph. These are the only two types of canonical subgraphs possible.

Canonical subgraphs are easily distinguishable by properties of their critical nodes. So by theorem 2, we can find all canonical subgraphs in $O(\log\log n)$ time using $n/\log\log n$ processors.

Theorem 3 (Monotonicity Theorem) *Given a* $D(i,u)$ *graph with a monotone list of weights* $w_i < w_{i+1} < \cdots < w_{u+1}$ *then the shortest path from* $(0,0)$ *to* (i,u) *is along the straight unit path* $(0,0) \nearrow (i,i) \rightarrow (i,i+1) \rightarrow \cdots \rightarrow (i,u)$.

This theorem also holds for a monotone list of weights having the relation $w_i > w_{i+1} > \cdots > w_{j+1}$ where the shortest path is $(0,0) \nearrow (j,j) \uparrow (j-1,j) \uparrow \cdots \uparrow (i,j)$.

Corollary 1 *If* $D(i,t)$ *contains no critical nodes, then there are no jumpers in a shortest path from* $(0,0)$ *to* (i,t). *Moreover, if* $D(i,t)$ *contains no critical nodes then the shortest path from* $(0,0)$ *to* (i,t) *is a straight unit path.*

As in [6] let $\|w_i : w_k\| = \sum_{j=i}^{k-1} w_j w_{j+1}$, which is easily computable using differences of components of one parallel prefix minima. The unit path $(i,j) \rightarrow \cdots \rightarrow (i,k)$ costs $w_i \| w_{j+1} : w_{k+1} \|$.

4 A Parallel Approximation Algorithm for the MCOP

Next we give an approximation algorithm for the MCOP to within 15.5% of optimality[2]. In addition, the processor time product of this algorithm is linear.

Given an instance of the MCOP with the weight list $l_1 = w_1, w_2, \ldots w_{n+1}$ and cyclically rotating it getting l_2 then finding an optimal parenthesization for l_2 gives an optimal solution to the original instance of the MCOP with l_1, see Hu and Shing [8]. So in the *rest of this paper* let w_1 denote the smallest weight in any weight list.

For the next theorem let w_i, w_{i+1}, and w_{i+2} be three adjacent weights in a weight list of an instance of the MCOP where $w_{i+1} > \max\{w_i, w_{i+2}\}$.

Theorem 4 (Chin [3]; and Hu and Shing [7]) *If*

$$w_1 w_i w_{i+2} + w_i w_{i+1} w_{i+2} < w_1 w_i w_{i+1} + w_1 w_{i+1} w_{i+2} \tag{1}$$

then the product $(a_i \bullet a_{i+1})$ is in an optimal parenthesization.

If equation 1 holds, then $w_{i+1} > \max\{w_i, w_{i+2}\}$. Therefore an ANSV match may not represent a minimal parenthesization in the MCOP, but it does provide a good approximation.

A list of weights is *reduced* iff for all weights, say w_{i+1}, with ANSV match $[w_i, w_{i+2}]$ equation 1 fails to hold, [3]. A reduced weight list may be non-monotonic.

Suppose in solving the ANSV problem w_j has match $[w_i, w_k]$ then generalize equation 1,

$$w_1 w_i w_k + w_i w_j w_k < w_1 w_j w_k + w_1 w_i w_j \tag{2}$$

and if products $(a_i \bullet \cdots \bullet a_{j-1})$ and $(a_j \bullet \cdots \bullet a_{k-1})$ are both in an optimal parenthesization then the product $(a_i \bullet \cdots \bullet a_{j-1}) \bullet (a_j \bullet \cdots \bullet a_{k-1})$ is also in an optimal parenthesization.

Given the weight list $l_1 = w_1, w_2, \ldots, w_{n+1}$ the approximation algorithm is [3, 7]: First reduce the weight list l_1 giving the weight list l_2, renumbering l_2 to be $l_2 = w_1, w_2, \ldots, w_{r+1}$ where $w_1 = \min_{1 \le i \le r+1}\{w_i\}$. Next if l_2 has more than two weights, then compute the depths of the parentheses for the linear product $((\cdots(a_1 \bullet a_2) \bullet \cdots) \bullet a_r)$ costing $w_1 \| w_2 : w_{r+1} \|$. With this, make the parenthesis discovered in step 1 the appropriate amount deeper. The depth of the parentheses determines the order to multiply the matrices.

Given the weight list l_1 the next algorithm outputs a reduced weight list. Let $A[1..n+1]$ be an array of $n+1$ integers all initialized to zero.

1. Solve the ANSV problem on the weight list l_1. Next check to see if there are any weights satisfying the condition described by equation 1. If there are none, then output l_1 and stop.

2. For all weights w_j in l_1 that have matches, say $[w_i, w_k]$, if w_j and w_i, w_k, satisfy equation 2, then assign a 1 to $A[j]$.

3. Now solve the ANSV problem on $A[1..n+1]$. If the nearest smaller values of $A[j]$ are in the match $[A[i], A[k]]$ then $(a_i \bullet \cdots \bullet a_{k-1})$ is in an optimal parenthesization. Removing all of the weights isolated by optimal parenthesizations gives a reduced weight list, which is output.

This algorithm produces a reduced weight list in $O(\log\log n)$ time with $n/\log\log n$ processors. Reducing a weight list in this way essentially uses the greedy principle since equations 4 and 2 find optimal substructures that are in approximately optimal superstructures.

---

[2] Similar results to those in this section were also very recently and independently reported by A. Czumaj in [5].

Theorem 5 (Chin [3]; and Hu and Shing [7]) *If a weight list $w_1, w_2, \cdots, w_{r+1}$ is reduced then the MCOP can be solved to within a multiplicative factor of* 1.155 *from optimal in constant time using n processors. This is done by choosing the linear parenthesization* $((\cdots(a_1 \bullet a_2) \bullet \cdots) \bullet a_r)$.

This algorithm shows that the path of critical nodes in the canonical subgraphs supply a good approximate solution for the matrix chain ordering problem.

5 Solving the Matrix Chain Ordering Problem in Parallel

This section gives a polylog time algorithm for solving the matrix chain ordering problem using $n^3/\log n$ processors.

In addition, assume that D_n contains critical nodes, otherwise by corollary 1 there is an immediate exact solution. Canonical subgraphs of the form $D^{(i,j)}$ are written as $D^{(1,m)}$ where $1 \leq m \leq n$. All of the results in this section apply to $D^{(i,t)}_{(j,k)}$ in addition to $D^{(1,m)}$ graphs.

5.1 Shortest Paths in Canonical Subgraphs

The first result of this subsection is theorem 6 which states that shortest paths containing no critical nodes are straight unit paths. A path q with one jumper, say $(i,j) \Longrightarrow (i,k)$, contains no critical nodes iff there are no critical nodes in q and there is a minimal path to $(j+1,k)$ without critical nodes. This generalizes to paths with more than one jumper. The second result of this subsection, theorem 7, considers the contributions of critical nodes to shortest paths in canonical subgraphs. Shortest paths between critical nodes have a characteristic structure.

Theorem 6 *Any shortest path q to some vertex (i,j) in $D^{(1,m)}$ where q contains no critical nodes, except possibly (i,j), is a straight unit path.*

Suppose (j,k) and (i,t) are two critical nodes in a canonical graph $D^{(1,m)}$, where $i \leq j \leq k \leq t$, there are two important symmetric paths between (j,k) and (i,t).
The *upper angular* and *lower angular* paths of (j,k) and (i,t) are

$$(j,k) \Uparrow (i,k) \rightarrow \cdots \rightarrow (i,t) \text{ and } (j,k) \Longrightarrow (j,t) \uparrow \cdots \uparrow (i,t)$$

respectively. For example see Figure 4.

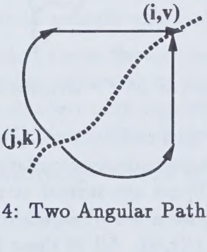


Figure 4: Two Angular Paths

Theorem 7 *In a canonical subgraph the shortest path between any two critical nodes that contains no other critical nodes is an angular path.*

Next we sketch a polylog time algorithm for finding shortest paths to all critical nodes in $D^{(1,m)}$ graphs. This algorithm takes $O(\log^2 m)$ time and uses $m^3/\log m$ processors.

First compute all of the unit paths to nodes in p in constant time using m processors. Perhaps, we can best view these paths as edges from $(0,0)$ to the nodes in p. The cost of each of the $O(m^2)$ angular paths can be computed in constant time using m^2 processors. Now compute the shortest path to each node in p by treating every angular path as a weighted edge and applying a parallel $(\min, +)$ matrix multiplication all pairs shortest path algorithm.

5.2 Combining the Canonical Graphs for an Efficient Parallel Algorithm

This subsection shows how to find shortest paths in entire D_n graphs. This is done by exploiting an underlying tree structure connecting the canonical subgraphs. At the same time we can treat these subgraphs atomically while ignoring the effect of the monotone subgraphs. Hence computing a shortest path is done by applying tree contraction techniques with special "leaf raking" operations.

In a D_n graph the structure joining all of the critical nodes is a *canonical tree.* Initially, for every canonical subgraph $D^{(i,j)}$ the critical node (i,j) is a leaf and is denoted by $\overline{(i,j)}$. Internal tree nodes are either isolated critical nodes or $\overline{(i,t)}$ and $\overline{(j,k)}$ for $D^{(i,t)}_{(j,k)}$ where $k \neq j+1$.

The edges of a canonical tree are the straight unit paths that connect tree nodes. Jumpers may reduce the cost of tree edges. An edge from $\overline{(i,j)}$ to $\overline{(i,k)}$ is denoted by $\overline{(i,j)} \Longrightarrow \overline{(i,k)}$ and all edges are directed towards $(1,n)$.

Theorem 8 (Hu and Shing [8], Yao [14]) *Given a weight list $w_1, \ldots, w_{n+1}$ with the three smallest weights w_1, w_{j+1}, and w_{k+1} then the products $w_1 w_{k+1}$ and $w_1 w_{j+1}$ are in some associative product(s) in an optimal parenthesization.*

There may be one or two fs that contain the products $w_1 w_{k+1}$ and $w_1 w_{j+1}$.

Corollary 2 (Atomicity Corollary) *Given a weight list $w_1, \ldots, w_{n+1}$ with the three smallest weights w_1, w_{j+1}, and w_{k+1} then the critical nodes $(1,j)$ and $(1,k)$ are in a shortest path from $(0,0)$ to $(1,n)$ in D_n.*

We can recursively apply this corollary.

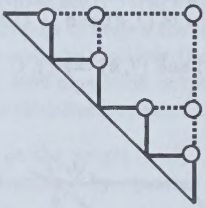


Figure 5: A Canonical Tree of $D^{(1,m)}$ Graphs, the Circles Denote Tree Nodes

Say D_n has fewer than $n-1$ critical nodes, then D_n may have several disconnected canonical trees and one or more monotone subgraphs by corollary 1. But, there is at least one path that joins these canonical subtrees. There are several structures that $D^{(1,m)}$ graphs may form. Let $\overline{(i,j)}, \overline{(j+1,k)}, \ldots, \overline{(u+1,v)}$ be a list of adjacent leaves in $D(i,v)$ where none of them is in a canonical tree outside of $D(i,v)$. All of these leaves together can share two possible relationships, or combinations of them: 1. The leaves are monotonically related so there are no internal tree nodes connecting them, for instance $w_i < w_{j+1} < \cdots < w_{u+1} < w_{v+1}$ or 2. The leaves form a binary canonical tree, for example see Figure 5. Here there are internal tree nodes in $D(i,v)$ connect the leaves together. We can solve the first case by creating a surrogate canonical tree and treating it as in case 2. Also, if $w_i = w_{j+1} = \cdots = w_{u+1} = w_{v+1}$ then we can handle this as an instance of case 2. The viability of this is shown by induction.

Theorem 9 *Given a tree node* $\overline{(i,u)}$ *where the graph* $D(i,u)$ *contains the tree nodes* $\overline{(i,j)}$ *and* $\overline{(j+1,u)}$. *Say* $D^{(j+1,u)}$ *is a canonical subgraph then a shortest path from* $\overline{(i,j)}$ *to* $\overline{(i,u)}$ *can be found in* $O(u-j)$ *operations.*

The symmetric case where $D(j+1,u)$ is a subgraph, and $D^{(i,j)}$ is a canonical subgraph follows. Finding the minimal cost of an edge in a canonical tree as described in theorem 9 is *edge minimization.* In parallel we can minimize all edges in a canonical tree in $O(\log n)$ time with $n/\log n$ processors since there are at most $n-1$ critical nodes in D_n.

This last theorem holds for leaves in the canonical tree that, after raking, have become conglomerates of other leaves and internal nodes. In this situation, jumpers derived from critical nodes in different subtrees are independent and compatible so we can minimize tree edges with them simultaneously. Edge minimizing unit edges in $D^{(i,t)}_{(j,k)}$ graphs is dealt with in the next subsection.

5.3 Contracting a Canonical Tree

This subsection contains a special rake operation for a standard tree contraction algorithm [10]. This completes an efficient solution to the MCOP. During tree contraction we consider both $D^{(1,m)}$ and $D^{(i,t)}_{(j,k)}$ canonical subgraphs.

Initially, all internal tree nodes have $mp(i,j)=\infty$ and all critical nodes in tree leaves have the minimum value from $(0,0)$ to (i,j) stored in $mp(i,j)$. Compute these values using the methods at the end of subsection 5.1. Also, each tree edge $\overline{(i,j)} \Longrightarrow \overline{(i,k)}$ initially weights $w_i \| w_{j+1} : w_{k+1} \|$.

For the rake operation, there is an ordering of the leaves that prevents the simultaneous raking of two adjacent leaves. Given two neighboring canonical graphs $D^{(i,j)}$ and $D^{(j+1,k)}$ with the three tree leaves $\overline{(i,j)}, \overline{(j+1,k)}$ and $\overline{(i,k)}$, say $w_i < w_{k+1} < w_{j+1}$ then leaf $\overline{(j+1,k)}$ must be raked since $\overline{(i,j)}$ is in a shortest path from $(0,0)$ to $\overline{(i,k)}$ by the Atomicity Corollary (corollary 2). Use the Euler Tour Technique [10] when the raking order is arbitrary.

Canonical graphs of the form $D^{(i,v)}_{(j,u)}$ can be nested in the sense that $D(j,u)$ may contain another canonical subgraph $D^{(k,t)}_{(r,s)}$ and so on. Given two nested canonical graphs $D^{(i,v)}_{(j,u)}$ and $D^{(k,t)}_{(r,s)}$, where $j \le k < t \le u$, and no such canonical subgraph is between them, then we can join them with one rake operation costing $O(\log^2 n)$ time and using $n^3/\log n$ processors.

In general we can join two nested canonical graphs of the form $D^{(i,v)}_{(j,u)}$ and $D^{(k,t)}_{(r,s)}$ as follows. First, throughout the whole D_n graph contract all canonical trees consisting of $D^{(1,m)}$ graphs as in subsection 5.2. Now edge minimize all straight unit paths in $D^{(i,v)}_{(j,u)}$ with jumpers that get their "*mp* values" from critical nodes that are in the subtrees between $D^{(i,v)}_{(j,u)}$ and $D^{(k,t)}_{(r,s)}$ that were just contracted. Next take all angular paths connecting these nested canonical subgraphs and apply a parallel $(\min,+)$ all pairs shortest path algorithm. With this step we can apply our extensions to the usual tree contraction algorithm completing the MCOP algorithm.

Theorem 10 (Tree Contraction Theorem) *We can solve the MCOP in* $O(\log^3 n)$ *time using* $n^3/\log n$ *processors.*

6 Conclusion

In terms of algorithm design paradigms, section 4 gives a greedy approximation algorithm. It is greedy in the sense that it isolates certain locally optimal matrix products. Next in section 5 the principle of optimality comes into play since the canonical graphs are well-formed substructures that may be in an optimal superstructure. In addition, the tree contraction algorithm gives an optimal matrix chain ordering product by combining such well-formed optimal substructures.

7 Acknowledgments

I would like to thank Alok Aggarwal, Gregory J. E. Rawlins and Greg Shannon for helpful comments and discussions.

References

[1] O. Berkman, D. Breslauer, Z. Galil, B. Schieber and U. Vishkin: "Highly Parallelizable Problems", *Symposium on the Theory on Computing*, 309-319, 1989.

[2] P. G. Bradford: "Efficient Parallel Dynamic Programming", Submitted.

[3] F. Y. Chin: "An $O(n)$ Algorithm for Determining Near-Optimal Computation Order of Matrix Chain Products", *Communications of the ACM*, Vol. 21, No. 7, 544-549, July 1978.

[4] T. H. Cormen, C. E. Leiserson and R. L. Rivest: *Introduction to Algorithms*, McGraw Hill, 1990.

[5] A. Czumaj: "An Optimal Parallel Algorithm for Computing a Near-Optimal Order of Matrix Multiplications", To Appear in *Proceedings of the 3rd Scandinavian Workshop on Algorithm Theory*, 1992.

[6] T.C. Hu and M.T. Shing: "Some Theorems about Matrix Multiplication", *Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science*, 28-35, 1980.

[7] T.C. Hu and M.T. Shing: "An $O(n)$ Algorithm to Find a Near-Optimum Partition of a Convex Polygon", *J. of Algorithms*, Vol. 2, 122-138, 1981.

[8] T.C. Hu and M.T. Shing: "Computation of Matrix Product Chains. Part I", *SIAM J. on Computing*, Vol. 11, No. 3, 362-373, 1982.

[9] T.C. Hu and M.T. Shing: "Computation of Matrix Product Chains. Part II", *SIAM J. on Computing*, Vol. 13, No. 2, 228-251, 1984.

[10] R. M. Karp and V. Ramachandran: "Parallel Algorithms for Shared Memory Machines", Chapter 17 in *Handbook of Theoretical Computer Science, Vol. A, Algorithms and Complexity*, V. Van Leeuwen—editor, Elsevier, 1990.

[11] M. Marcus: *Introduction to Modern Algebra*, Marcel Dekker, 1978.

[12] W. Rytter: "On Efficient Parallel Computation for Some Dynamic Programming Problems", *Theoretical Computer Science*, Vol. 59, 297-307, 1988.

[13] L. G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff: "Fast Parallel Computation of Polynomials Using Few Processors", *SIAM J. on Computing*, Vol. 12, No. 4, 641-644, Nov. 1983.

[14] F. Frances Yao: "Speed-Up in Dynamic Programming", *SIAM J. on Algebraic and Discrete Methods*, Vol. 3, No. 4, 532-540, 1982.

Parallel Algorithms for Convex Bipartite Graphs and Related Problems

M. G. Andrews and D. T. Lee
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208

Abstract

We consider the problem of finding a maximum matching and a maximum independent set of vertices in a convex bipartite graph. A bipartite graph $G = (A, B, E)$ is convex on the vertex set B if B can be ordered so that for each element a in the vertex set A the elements of B connected to a form an interval of B.

We give a parallel algorithm of time complexity $O(\log^2 n)$ and processor complexity $O(n)$ based on the hypercube model for the maximum matching. We give an $O(\log^2 n)$ time algorithm using $O(n/\log n)$ processors on the CREW-PRAM for the MIS problem; on the hypercube, the time complexity is $O(\log^3 n)$ using $O(n)$ processors. We also show that the range minima and lowest common ancestor problems can be solved in $O(\log^2 n)$ time using $O(n)$ processors on the hypercube.

Introduction

$F \subseteq E$, is a matching in a graph, G, iff no two edges in F have a common endpoint. F is a maximum cardinality matching in G iff (a) F is a matching and (b) G contains no matching, H, such that $|H| > |F|$. ($|H|$ is the number of edges in H). A maximum matching in Figure 2 is shown by the dashed lines.

A maximum independent set (MIS) in an undirected graph is a maximum collection of vertices, I, such that no pair of vertices in I is adjacent. For general graphs, a parallel deterministic algorithm that solves the maximal independent set problem in $O((\log n)^4)$ time using $O(n^3/(\log n)^3)$ PEs in the EREW-PRAM model is given in [KW].

A convex bipartite graph, G, is a triple (A, B, E). $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ are disjoint set of vertices. E is the edge set which has the property that for each element a in the vertex set A the elements of B connected to a form an interval of B.

In the triple representation of convex bipartite graphs, G is uniquely represented by the set of triples: 1) $T = \{(i, s_i, h_i) | 1 \leq i \leq n\}$, 2) $s_i = \min\{j | (a_i, b_j) \in E\}$, and 3) $h_i = \max\{j | (a_i, b_j) \in E\}$. In this representation, we explicitly record the smallest (s_i) and the highest (h_i) index vertices to which each a_i is connected.

Dekel and Sahni [DS], give a $O(\log^2 n)$ time and $O(n)$ PE parallel algorithm for maximum matching in convex bipartite graphs using the Shared Memory Model (SMM).

Related to the maximum matching problem in convex bipartite graphs is the determination of a maximum independent set (MIS) of vertices. A MIS can be derived from a maximum matching, F, by using standard alternating path techniques to find all vertices of G reachable from the unmatched vertices of A. Lipski and Preparata [LP] used the maximum matching of a convex bipartite graph to derive the MIS of a convex bipartite graph in $O(n+m)$ sequential time. The standard alternating path technique they used is highly sequential and does not map directly to a parallel architecture.

In this paper, we show that a maximum matching for a convex bipartite graph can be obtained in time $O(\log^2 n)$ with $O(n)$ PEs on the hypercube model. We also show that the MIS can be determined in $O(\log^2 n)$ time using $O(n/\log n)$ PEs on the CREW-PRAM model using the maximum matching results. The parallel algorithm on the hypercube to generate the MIS from the maximum matching takes $O(\log^3 n)$ time using $O(n)$ PEs.

Range Minima Algorithm

In this section, we provide an algorithm for the range minima primitive on the hypercube model. This will be used later for the MIS problem.

The Range Minima problem: Given an array of n real numbers $A[1\dots n] = (a_1, a_2, \dots, a_n)$, preprocess A. After the preprocessing, we are given a set of queries (subarrays), $A[l(i)\dots r(i)]$, of size m, $1 \le i \le m \le n$. For each of these queries, we want to compute the minimum element.

The Range Minima problem preprocessing can be solved optimally in $O(\log\log n)$ time using $O(n/\log\log n)$ processors on a CRCW PRAM [BBGSV] or in $O(\log n)$ time using $O(n/\log n)$ processors on an EREW PRAM [Ki]. On a PRAM, one processor can find in constant time the minimum element in a subarray. On a CREW-PRAM model, m queries can be processed by m processors, also in $O(1)$ time. On a hypercube, simulation of a PRAM algorithm takes additional $O(\log^2 n)$ time with $O(n)$ processors. However, by using specialized search structures, we can improve the time complexity of the preprocessing step.

In order to implement the range-minima algorithm on the hypercube, the data structure will be that of a complete binary tree whose leaves are the elements of the array. The value at each internal node will be the minimum of the elements in the leaves of the subtree rooted at that node, similar to the method of [GR]. For example, if the leaves of the tree are indexed from left to right, respectively, by the integers, 1 to 8, then (for $i \le j$ and $1 \le i, j \le 8$) $\min[i\dots j]$ is the minimum of the elements stored in the interval $[i\dots j]$. Each interval at a node of the tree is a *standard* interval. Other intervals are *non-standard.* This data structure can be built in $O(\log n)$ time.

For a non-standard interval, we use the balanced binary tree of standard intervals as a substrate for the computation. We begin with the non-standard interval at the root and then divide it into one or two intervals using the same division point as that of the root interval into its child intervals. The interval(s) obtained from the non-standard interval are then passed to the children of the root. These interval(s) have their (at most one in each case) limits modified in an obvious way. Then the process is repeated. Figure 1 shows the process in a schematic manner for a particular example. If the limits of an interval generated by this process coincide with the limits of the standard interval at the current node of the binary tree, then no further partitioning of this interval is required. In this way, a set of at most $\log n$ intervals (into which the original non-standard interval is partitioned) is built up. In Figure 1, P_r and P_l denote the two paths which are traced from the root in this process and off which there are spurs (branch paths of length 1) to standard intervals belonging to the partition. The length of paths P_r and P_l are $O(\log n)$. For each node that we visit on the path P_l whose right child (RC) standard interval lies within the query range, we check the value of its minimum. Likewise, for each node on P_r, we check the left child (LC). Then the minima for the query is the smaller of the minima found on the path P_l or on the path P_r. Note that at each level in the tree, we spend constant time, so that for a single query, the computation takes $O(\log n)$ time.

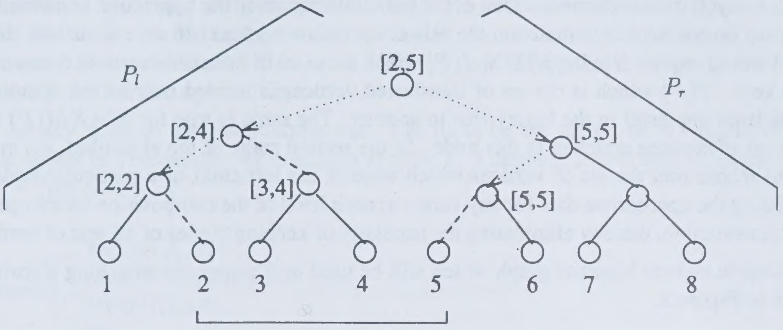


Figure 1: Illustration of Range Minima Computation

For multiple queries, we convert the binary tree to a search structure according to the method of [DR]. In $O(1)$ time, each processor can compute its location, j, in the search structure. As part of the search structure, for each node, we construct two records which contain the locations of its left and right child and also the value of the minima for each of its children. The search structure is then built by routing, for each node, information from the node in the binary tree to the node j. This routing step will take $O(\log^2 n)$ time.

For a range query (i, j), we first determine in $O(1)$ time (we can do this because it is a complete binary tree), $v = LCA(i, j)$. Then let $u = LC(v)$ and $w = RC(v)$. Let $i1$ be the right most leaf in the subtree rooted at u. Let $j1$ be the left most leaf in the subtree rooted at w. We then create two queries for the search structure, Range-minima$(i, i1)$, and Range-minima$(j1, j)$. For the left (resp., right) range, the query starts at u (resp., w). As the search goes from u (resp., w) to i (resp., j), we select the minima of all RC (resp., LC) of each node on the path from u (resp., w) to i (resp., j) as long as the range of the leaves in the subtree rooted at the RC (resp., LC) is within $i \ldots i1$ (resp., $j1 \ldots j$). In the [DR] algorithm, all queries must begin at the root. We simply pass the query along the appropriate path from the root to the start node.

From the [DR] algorithm, for a search structure of height h, we know that the query processing will take $O(h \log n) = O(\log^2 n)$ time. Therefore, we have the following theorem.

Theorem 1 The range minima problem can be solved for an arbitrary number, m, $(m \leq n)$ of queries in time $O(\log^2 n)$ on a hypercube of size $O(n)$ after a preprocessing step of time complexity $O(\log^2 n)$. After the preprocessing, a single query can be solved in $O(\log n)$ time.

Maximum Matching on Convex Bipartite Graphs

In this section, we present an $O(\log^2 n)$ time algorithm with $O(n)$ processors on a $(\log_2 n)$-dimensional hypercube. In general, we use the same algorithm as [DS]. The computational tree for the algorithm is a complete binary tree which maps in a straightforward manner to the hypercube architecture. We maintain the data elements (A vertices) separate from the computational tree. In the computational tree, we keep, on a per-node basis, pointers to that node's array(s) of data elements. One of the basic differences in the hypercube implementation s that we do not explicitly maintain the sets of candidate vertices. If we maintained all these sets, it would require $O(n \log n)$ PEs. $I(P)$ which is the set of infeasible vertices does not need to be kept. $T(P)$ which is the set of transferred vertices is needed only as the computation moves from one level in the binary tree to another. The same is true for $MAXM(P)$ which is the set of vertices matched at this node. In the second stage of the algorithm, we must be able to reconstruct the set of vertices which were at the left child of a particular node. By populating the appropriate data for any vertex at each level of the computation we can perform the reconstruction, thereby eliminating the necessity of keeping copies of all sets of vertices.

An example convex bipartite graph which will be used to illustrate the matching algorithm is shown in Figure 2.

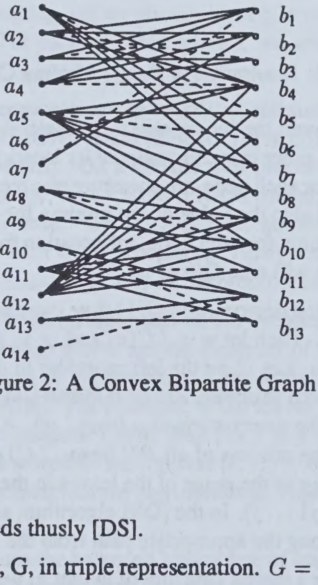


Figure 2: A Convex Bipartite Graph

Description of Algorithm

In general, the algorithm proceeds thusly [DS].

Input: A convex bipartite graph, G, in triple representation. $G = \{i, s_i, h_i | 1 \leq i \leq n\}$.

Output: $MTCH(i)$, for each point, i, in A, the point in B to which it is matched or null.

Step 1: The points of A are sorted by s_i and within s_i by h_i. The distinct s_i values are then identified. Let these be $R_1, R_2, \ldots, R_k$.

Step 2: Assume that $R_1 < R_2 < \ldots < R_k$. Then a complete binary tree is formed with 2^l leaf nodes, $2^{l-1} < k \leq 2^l$. With each node, P, in the tree, we associate a contiguous subset $\{u, u+1, u+2, ..., v\}$ of the vertices of B. This subset is denoted $[u, v].P$, or simply $[u, v]$. Figure 3 shows the vertices of A after sorting.

PE	0	1	2	3	4	5	6	7	8	9	10	11	12	13
i	7	3	2	4	6	1	5	12	9	8	11	14	10	13
s_i	1	1	1	1*	4	4	4	4*	9	9	9*	12	12	12*
h_i	2	3	4	4	4	8	10	11	10	11	13	12	13	13

Figure 3: Sorted Vertices of A

Only the first k leaf nodes will be populated. Let these nodes be numbered 1 through k, left to right. If P is the ith leaf node, then $[u, v].P$ is $[R_i, R_{i+1} - 1]$. If P is not a leaf node, the the subset of B associated with P is $[u, v].L(P) \cup [u, v].R(P)]$, where $L(P)$ and $R(P)$ are, respectively, the left and right children of P.

Let P be any node of the computation tree. Let $[u, v]$ be the subset of B associated with P. The subset of A available for matching at node P is denoted $M(P)$ and is defined to be $M(P) = \{i | u \leq s_i \leq v\}$.

For example,
M(1)=1,2,...,14
M(2)=7,3,2,4,6,1,5,12
M(4)=7,3,2,4
etc.

The subset, $M(P)$, will be partitioned into three subsets, $MAXM(P)$, $I(P)$, and $T(P)$ as a result of the computation. $MAXM(P)$ is a maximum cardinality subset of $M(P)$ that may be matched with vertices in $[u, v].P$. This subset is called the matched set. $I(P)$ denotes the infeasible set; it consists of all vertices $i \in (M(P) - MAXM(P))$ such that $h_i \leq v$. The transferred set, $T(P)$, consists of all vertices $i \in (M(P) - MAXM(P))$, such that $h_i > v$ (i.e., it may find a match when nodes are combined).

Two passes are made over the binary tree. The first pass begins at the leaves and moves toward the root. During this pass, the sets $MAXM$, I, and T for each node are computed. The second pass starts at the root and progresses towards the leaves. In this pass, the $MAXM$ set of each node is updated so as to correspond to the set of A vertices matched to the B vertices associated with that node.

Step 3: For the first pass, within a node, the value of s_i is not considered. Matchings are determined based on the highest index, h_i. For a leaf node, $MAXM$, I, and T are computed using the subset of vertices of A as the input to procedure $FEAS$ (see [DS] for details). For a non-leaf node, $MAXM$, I, and T are computed using the $MAXM$, I, and T sets of the left and right children. The input set to procedure $FEAS$ at a non-leaf node is $T(L) \cup MAXM(R)$. (Since both of these sets are sorted, they are simply merged together). Let S be the subset of $T(L) \cup MAXM(R)$ that is matched by $FEAS$. Then $MAXM(P) = MAXM(L) \cup S$. (This also is a simple merge in order of h_i). $T(P)$ is the set of vertices that are not matched by $FEAS$ and for whom $h_i > v_R$. Since $I(P)$ is never used, it is not necessary to keep track of it.

For pass 2 of the algorithm, we must be able to obtain the set $MAXM(L)$ for all left children. To do this, we observe that $MAXM(L)$ is a subset of $MAXM(P)$ (and $MAXM'(P)$) for all P. For all $a_i \in A$, we maintain the data item $LFT(i)$. In pass 1, for each level, j, in the binary tree, we set the jth bit in $LFT(i)$ equal to 1, for all $a_i \in MAXM(L)$ (for all left children at this level of the computation).

Step 4: In the second pass, for each vertex P of the binary tree, we compute a set $MAXM'(P)$ which represents the set of A vertices matched with the set $[u, v].P$. If P is the root node, then

$MAXM'(P) = MAXM(P)$, by definition of $MAXM(P)$. Let P be any non-leaf node for which $MAXM'(P)$ has been computed. Let L and R be the left and right children, respectively, of P. Let $[u,v].L = [u_L, v_L]$ and $[u,v].R = [u_R, v_R]$. Let $V = \{j | j \in MAXM'(P)$ and $s_j < u_{L(P)}\}$. For level j, to derive $MAXM(L)$ from $MAXM'(P)$, we simply select $a_i \in MAXM'(P)$ where the jth bit of $LFT(i) = 1$. Let W be the ordered set obtained by merging together V and $MAXM(L)$. Then $MAXM'(L)$ consists of the first $min\{|W|, (v_L - u_L + 1)\}$ vertices in the set W. For the right child, $MAXM'(R) = MAXM'(P) - MAXM'(L)$.

Step 5: For the $MAXM'()$ sets of the leaves, the matching is obtained as follows: The first vertex in $MAXM'(P)$ is matched with u, the second with $u + 1$, etc.

Complexity Analysis

Step 1 takes $O(\log^2 n)$ time for sorting.

In step 2 distinct s_i values (denoted $R_1, R_2, \ldots, R_k$) are identified and sent to leaf nodes in the binary tree using Rank, Concentrate, and Distribute. For each node in the binary tree, the sets $[u,v].P, [As, Ae].P, [MAXMs, MAXMe].P$, and $[Ts, Te].P$, are obtained using parallel prefix and Rank. As (resp., Ae) denotes the index of the start (resp., last) PE in the string of PEs where the vertices of A for this node are stored. Similar notation holds for $MAXM$ and T.

It is important to note that as we go up the binary tree, at any non-leaf node we are always working with adjacent strings of processors. Therefore, it is a simple matter to adjust the range of processors associated with the string for that node. Then for a non-leaf node, when it is necessary to merge sets, the two substrings are merged into a larger string of processors using Rank and Translation and Merge. Each string of processors can independently perform operations such as Broadcast, Max, Merge, Rank, etc. [BM]. All operations in this step take $O(\log n)$ time.

For step 3, we use Distribute and Broadcast to make the relevant data available to all PEs. Rank, Concentrate and Translate are used to identify the elements in each set and manipulate them into "strings". The appropriate ranges are identified and updated using the operations, MAX, MIN, Concentrate, and Distribute.

When proceeding to the next level i, sets are combined using Translate and Merge. We also set the ith bit in $LFT(j)$ for all $j \in MAXM(L)$ in constant time. Step 3 is executed once for each level of the computational tree which is $\log k$, $k \leq n$ times. All operations in this step take at most $O(\log n)$ time.

Step 4: Now we are ready to start pass2. At this point, the root will have the set of A vertices which have been matched. However, the matching vertex of B has not yet been assigned. The set of matched A vertices is expressed as a range of processor indices, $[MAXMs, MAXMe].Root$. The set of candidate B vertices is $[u,v].Root$.

At each level i, we first derive all A vertices in $MAXM(L)$ by checking the ith bit in $LFT(j)$, $j \in MAXM'(P)$. We then broadcast $u_{L(P)}$ to all vertices $i \in MAXM'(P)$ for each node P at this level, (using Concentration, Distribution and Broadcasting within each string). Each processor, in parallel, checks if $s_i < u_{L(P)}$, and if so, sets $FLAG = 1$. Within a string, these are Concentrated and constitute the set, $V(P)$. Sets $V(P)$ and $MAXM(L)$ are Merged to form the set $W(P)$. Let $z = v_L - u_L + 1$. We select the first $\min\{|W|, z\}$ vertices in each string, $W(P)$, by using Broadcast and Rank. These are then Concentrated, and constitute the set, $MAXM'(L)$. The remaining vertices, $MAXM'(P) - MAXM'(L)$ constitute the set, $MAXM'(R)$.

We next update $[MAXM's, MAXM'e].L$ and $[MAXM's, MAXM'e].R.$, by using Rank, MAX, MIN, Concentrate, and Distribute operations. Step 4 is executed once for each level of the computational tree which is $\log k$, $k \leq n$ times. All operations in this step take at most $O(\log n)$ time.

Step 5: Finally, we are back to the leaf nodes in the binary tree. Each node has $[MAXM's, MAXM'e]$ corresponding to a range of processors each associated with a vertex of A. Also, the range of B vertices associated with this node is $[u, v].P$. Using Broadcast and Rank, we compute the vertex of B to be assigned to each vertex of $MAXM'(P)$.

Based on the above analysis, we have the following theorem.

Theorem 2 A maximum matching for a convex bipartite graph can be found in $O(\log^2 n)$ time using $O(n)$ processors on a hypercube.

Maximum Independent Set for Convex Bipartite Graphs

In this section, we present a parallel algorithm to compute the MIS for a convex bipartite graph, G.

Suppose the graph, G, is described by the arrays $s[i]$ and $h[i]$ and that $MATCH[b_j]$ gives for each $b_j \in B$ either Λ or the element of A matched to it. Assume that the vertices of A are ordered as they appear in the final step of the matching algorithm. Note that when A is in this order, the vertices of B in the matching are in increasing order. Let A_0 denote the set of vertices in A which are not matched. We now direct edges in the matching from $B \rightarrow A$ and edges not in the matching from $A \rightarrow B$. Due to the property of convexity, for each $a \in A_0$, the set $B(a)$, *i.e.*, the set of vertices reachable by a single edge from a, forms an interval of B. From any matched vertex, b, in this interval we reach a single vertex, $MATCH[b] \in A$ which in turn reaches another interval $B(MATCH[b])$ of B. Note that $B(a)$ and $B(MATCH[b])$ necessarily overlap, so that by the convexity of G, their union is a single interval. For each $a \in (A - A_0)$, the set of vertices reachable by a single edge from a, forms one or two intervals of B (call these upper and lower intervals). As for $a \in A_0$, for each $a \in (A - A_0)$, we use $MATCH[b]$ to extend the interval(s) reachable from a.

For each $a \in A$, the algorithm proceeds by constructing at each iteration, the one (or two) extended intervals of B, which is (are) the set of all elements of B reachable from a. By constructing the extended intervals for each $a \in A$ in parallel, we are essentially using path compression to reduce the number of interval extending operations for each $a \in A_0$ from n to $\log n$. Now, for each iteration of the algorithm, we want to extend the upper and lower interval of each $a \in A$ as far as possible in both directions. To do this, for each $a_i \in A$ we want to find the maximum b vertex and the minimum b vertex which can be reached by following any matched edge, $b_j \rightarrow MATCH(b_j)$, b_j in lower interval of a_i or b_j in upper interval of a_i.

Suppose that we have the convex bipartite graph of Figure 2. From the matching algorithm presented earlier, we get $A_0 = \{a_6, a_{13}\}$ and the matched vertices shown in Figure 4. If we now direct edges in the matching from $B \rightarrow A$ and edges not in the matching from $A \rightarrow B$, we get the upper and lower intervals shown. (s_u and h_u denote the range of the upper interval, s_l and h_l denote the range of the lower interval. For example, for a_5 and a_8, the upper and lower intervals are detailed in the Figure 5. For a_5, its upper interval is $[b_4 \dots b_5]$. From Figure 4 we see b_4 is matched to a_4 which has an interval of $[b_1 \dots b_4]$ and this will extend the interval the furthest on the upper side. a_5's lower interval is $[b_7 \dots b_{10}]$, these b-vertices are matched to a_{12}, a_9, and a_8. Of these three vertices, from Figure 4 we see either a_{12} or a_8 will extend the interval the furthest on the lower end (to b_{11}). Thus, the new upper interval of a_5 is $[b_1 \dots b_5]$; the new lower interval is $[b_4 \dots b_{11}]$. Since the upper and lower intervals now overlap, we combine them into the single interval $[b_1 \dots b_{11}]$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
a_i	a_7	a_3	a_2	a_4	a_6	a_1	a_5	a_{12}	a_9	a_8	a_{11}	a_{14}	a_{10}	a_{13}
s_i	b_1	b_1	b_1	b_1	b_4	b_4	b_4	b_4	b_9	b_9	b_9	b_{12}	b_{12}	b_{12}
h_i	b_2	b_3	b_4	b_4	b_4	b_8	b_{10}	b_{11}	b_{10}	b_{11}	b_{13}	b_{12}	b_{13}	b_{13}
matching b_i	b_1	b_2	b_3	b_4	–	b_5	b_6	b_7	b_9	b_{10}	b_{11}	b_{12}	b_{13}	–
s_u	–	b_1	b_1	b_1	b_4	b_4	b_4	b_4	–	b_9	b_9	–	b_{12}	b_{12}
h_u	–	b_1	b_2	b_3	b_4	b_4	b_5	b_6	–	b_9	b_{10}	–	b_{12}	b_{13}
s_l	b_2	b_3	b_4	–	–	b_6	b_7	b_8	b_{10}	b_{11}	b_{12}	–	–	–
h_l	b_2	b_3	b_4	–	–	b_8	b_{10}	b_{11}	b_{10}	b_{11}	b_{12}	–	–	–

Figure 4: Vertices of A with Mate and Intervals Indicated

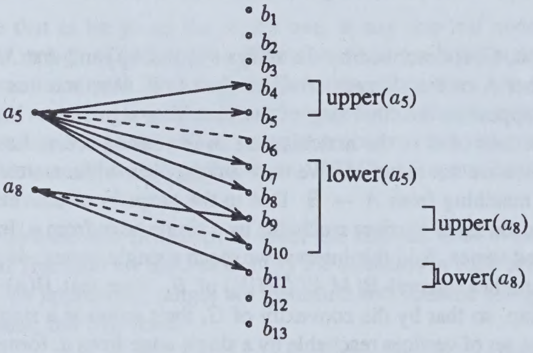


Figure 5: a_5 and a_8 detail

Suppose that we create a new data structure, $BMINMAX$, which is an array of vertices of B that are in the matching and the corresponding range $(s_{b_i} \dots h_{b_i})$ of vertices of B that can be reached by following the edge $(b_i \rightarrow MATCH(b_i))$. s_{b_i} and h_{b_i} are obtained from s_u and h_l values for $MATCH(b_i)$. Using $BMINMAX$, given an interval of B, say $[(b_j \dots b_k)]$, we compute its corresponding positions in $BMINMAX$, say (j', k'), then the interval can be extended to (b_y, b_z), where $b_y = \min(s_{b_i} | j' \leq i \leq k')$ and $b_z = \max(h_{b_i} | j' \leq i \leq k')$. Computation of b_y (resp., b_z) is the same as the Range Minima (resp., Maxima) Search problem. It is possible for one or more of the vertices which define $[s_u, h_u]$ and $[s_l, h_l]$ to not be matched. If that is the case, then before executing range search, we substitute for s_u (resp., s_l) the first matched b vertex below s_u (resp., s_l) and for h_u (resp., h_l) the first matched b vertex above h_u (resp., h_l).

For each iteration of the extending process, once we have computed the new upper and lower intervals for each $a \in A$, we then rebuild the data structure $BMINMAX$ and compute the

i	1	2	3	4	5	6	7	8	9	10	11	12
b_i	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_9	b_{10}	b_{11}	b_{12}	b_{13}
$MATCH(b_i)$	a_7	a_3	a_2	a_4	a_1	a_5	a_{12}	a_9	a_8	a_{11}	a_{14}	a_{10}
s_{b_i}	b_1	b_1	b_1	b_1	b_4	b_4	b_4	b_9	b_9	b_9	b_{12}	b_{12}
h_{b_i}	b_2	b_3	b_4	b_4	b_8	b_{10}	b_{11}	b_{10}	b_{11}	b_{11}	b_{12}	b_{13}

Figure 6: BMINMAX

new extended intervals. The iterations terminate when there is no change in the extended interval for any $a \in A$. Then, using the final extended intervals, we compute B_1 which is the set of all B vertices reachable from $a \in A_0$. $B_1 = \{b_j | s_{u_i} \leq j \leq h_{u_i}, i \in A_0\}$. Then we compute the set A_1 which is the mate of each vertex in B_1 union with the set A_0. $A_1 = \{MATCH(i) | b_i \in B_1\} \cup A_0$. Finally, the MIS is given by $MIS = A_1 \cup (B - B_1)$.

Time complexity analysis (for CREW-PRAM): Setting up the data structure for $BMINMAX$ can be done in $O(\log n)$ time with $O(n/\log n)$ PEs using parallel prefix and parallel postfix. Preprocessing of $BMINMAX$ for use in range query can be done in time $O(\log n)$ using $O(n/\log n)$ PEs [We]. Computation of the new extended intervals takes constant time on a CREW-PRAM. Computation of B_1, after all extended intervals have been computed can be done in constant time as can the computation of A_1 and MIS. Obviously, the main loop is executed no more than $O(\log n)$ times. Thus, we have the following theorem.

Theorem 3 On a CREW-PRAM, the time required to compute the MIS of a convex bipartite graph, $G(A, B, E), A = \{a_1, a_2, \ldots, a_n\}, B = \{b_1, b_2, \ldots, b_m\}$, is $O(\log^2 n)$ using $O(n/\log n)$ PEs, where $n = \max(m, n)$.

Time complexity analysis (for Hypercube): Setting up the data structure for $BMINMAX$ can be done in $O(\log n)$ time with $O(n)$ PEs using parallel prefix and parallel postfix and concentrate operations. Preprocessing of $BMINMAX$ for use in range query can be done in time $O(\log^2 n)$ using $O(n)$ PEs. Computation of the new extended intervals takes $O(\log^2 n)$ using $O(n)$ PEs. Combining intervals takes constant time. Computation of B_1, after all extended intervals have been computed can be done in $O(\log n)$ time for Merging as can the computation of A_1 and MIS. Since the main loop of the algorithm is executed no more than $(\log n)$ times, we have the following theorem.

Theorem 4 On a hypercube, the time required to compute the MIS of a convex bipartite graph, $G(A, B, E)$, $A = \{a_1, a_2, \ldots, a_n\}$, $B = \{b_1, b_2, \ldots, b_m\}$, is $O(\log^3 n)$ using $O(n)$ PEs, where $n = \max(m, n)$.

Lowest Common Ancestor

Wen has shown that the Lowest Common Ancestor (LCA) problem can be solved in $O(\log\log n)$ time using $O(n/\log\log n)$ PEs on the common CRCW PRAM or in $O(\log n)$ time using $O(n/\log n)$ PEs on an EREW PRAM, given an ordered tree along with its preorder and postorder traversals [We]. The preprocessing enables one PE to find in constant time the LCA of any pair of vertices in the tree. Wen's algorithm use the primitives, All Nearest Smaller Values (ANSV), All Closest Enclosing Parenthesis (ACEP), and Range Minima. ACEP finds for each matched pair of parenthesis in an array of well-formed parenthesis, the closest pair of parenthesis that encloses it. It has complexity $O(ANSV)$. [Ry] has shown that the ANSV problem can be solved in time $O(\log^2 n)$ with $O(n)$ PEs on the hypercube. Combining this result with the range minima result of Section 2, we have the following theorem.

Theorem 5 Given an ordered tree along with its preorder and postorder traversals, a preprocessing can be done in time $O(\log^2 n)$ on a hypercube of size $O(n)$. After the preprocessing, the lowest common ancestor of any pair of vertices in the tree can be found in $O(\log n)$ time. The LCA for an arbitrary number, m, $(m \leq n)$ of pairs of vertices can be found in time $O(\log^2 n)$ on a hypercube of size $O(n)$.

References

[BBGSV] O. Berkman, D.Breaslauer, Z. Galil, B. Schieber, and U. Vishkin, "Highly Parallelizable Problems", Proceedings of the Annual IEEE Symposium on Foundations of Computer Science", (1989) 309-319.

[BM] L. Boxer and R. Miller, "Dynamic Computational Geometry on Meshes and Hypercubes", *Proceedings International Conference on Parallel Processing*, (1988) 323-330.

[DR] F. Denhe and A. Rau-Chaplin, "Implementing Data Structures on a Hypercube Multiprocessor, and Applications in Parallel Computational Geometry", *Journal of Parallel and Distributed Computing*, **8**, (1990) 367-375.

[DS] E. Dekel and S. Sahni, "A Parallel Matching Algorithm for Convex Bipartite Graphs", Proceedings International Conference on Parallel Processing, (1982) 178-184.

[GR] A. Gibbons and W. Rytter, "Efficient Parallel Algorithms", Cambridge University Press, Cambridge, GB (1988).

[Ki] S. K. Kim, "Optimal Parallel Algorithms on Sorted Intervals", Proceedings of the 27th Annual Allerton Conference on Control, Communication, and Computing, (1989) 766-778.

[KW] R. M. Karp and A. Wigderson, "A Fast Parallel Algorithm for the Maximal Independent Set Problem", Proceedings of the 16th ACM Symposium on Theory of Computing, (1984) 266-272.

[LP] W. Lipski, Jr., and F. P. Preparata, "Efficient Algorithms for Finding Maximum Matchings in Convex Bipartite Graphs and Related Problems", *Acta Informatica*, **15**, (1981) 329-346.

[Ry] K. Ryu, "Efficient Parallel Algorithms on the Network Model", UMIACS-TR-90-37, CS-TR-2426, University of Maryland, March 1990.

[We] Z. Wen, "Some Doubly Logarithmic Time Optimal Parallel Algorithms", Proceedings Allerton Conference on Communication, Control, and Computing, (1991) 847-855.

Parallel Edge Coloring Approximation

Martin Fürer Balaji Raghavachari

Department of Computer Science
Pennsylvania State University
University Park, PA 16802

Abstract

Let G be a graph with n vertices and m edges and let its maximum degree be Δ. It is shown that a valid edge coloring of G using at most $2\Delta - 1$ colors can be computed in $O(\log n \log \Delta)$ time using $O(m+n)$ processors on a *CREW PRAM*. Using this, for any constant $c > 1$, a valid edge coloring for G using at most $c\Delta$ colors can be computed in $O(\log^2 n)$ time, using a linear number of processors. Using different techniques, we show that it is possible to compute a Δ^2 coloring in $O(\log^* n)$ time, with $O(m+n)$ processors. Also, a maximal matching of G can be computed in $O(\log^2 n \log \Delta)$ time with a linear number of processors on a *CREW PRAM*.

1 Introduction

Consider a graph $G = (V, E)$ with n vertices and m edges and let its maximum degree be Δ. An *edge coloring* of G is a coloring of its edges using the colors $\{1, 2, ..., k\}$ in such a way that no two edges incident on the same vertex receive the same color. The smallest k for which such a coloring is possible is known as the *chromatic index* ($\chi'(G)$) of G. It is obvious that $\chi'(G) \geq \Delta$, the degree of G. Vizing [20] has shown in a remarkable theorem that $\chi'(G) \leq \Delta + 1$. In other words, $\chi'(G)$ is either Δ or $\Delta + 1$. But Holyer [9] has shown that deciding whether the chromatic index is Δ is *NP*-complete. Therefore except in special cases like bipartite graphs and planar graphs of large degree, a $\Delta + 1$ coloring is generally accepted as practically optimal.

The edge coloring problem appears in many places (see for example [7], [8], [14], [15]) and is one of the fundamentally important problems in graph theory. The inductive proof given by Vizing leads to a sequential algorithm which produces a $\Delta + 1$ coloring of G in $O(mn)$ time. In the case of bipartite graphs, it is easy to show that $\chi'(G) = \Delta$. Algorithms for finding such a coloring efficiently have been given by Gabow and Kariv [5] and Cole and Hopcroft [3]. So far efficient parallel algorithms for solving the above problem have not been discovered except for restricted classes of graphs. Lev, Pippenger and Valiant [15] have demonstrated a parallel algorithm to compute a Δ-coloring for bipartite graphs in $O(\log^2 n \log \Delta)$ time using $O(m+n)$ processors. Karloff and Shmoys [12] showed that when Δ is bounded by a polylog in n, a $\Delta+1$ edge coloring of G can be computed in *NC*, which is the class of problems that can be solved in polylog time with a polynomial number of processors. For graphs of constant degree, their algorithm can be implemented to run in $O(\log^2 n)$ time with $O(n)$ processors. Linial [16] has devised an algorithm for computing a Δ^2 coloring in $O(\log^* n)$ time. But his algorithm is based on the existence of certain sets, and it is not known how to construct these sets. In essence he shows that such an algorithm exists and not how to find it.

In this paper we look at parallel approximation algorithms for this problem. Our goal is to be able to color the edges of the given graph efficiently in parallel in such a way that the number of colors used is close to optimal. Such an algorithm would be useful in other parallel algorithms which use edge coloring as a subroutine. Our technique was used to save a factor of $O(\log^2 n)$ from the running time of computing Hamiltonian Cycles in a special class of dense directed graphs [4].

The following is a summary of the results in this paper. We first show how to color the edges of a graph using at most $2\Delta - 1$ colors in $O(\log n \log \Delta)$ time. We show how to extend some of the ideas to compute a coloring which uses at most $c\Delta$ colors, for any given constant $c > 1$, in $O(\log^2 n)$ time. Using different techniques, we show that it is possible to compute a Δ^2 coloring in $O(\log^* n)$ time. We also demonstrate that a maximal matching of G can be computed in $O(\log^2 n \log \Delta)$ time. This improves the running time of the best deterministic algorithm for maximal matchings, which was given by Israeli and Shiloach [11]. It runs in $O(log^3 n \log \Delta)$ time on the *CREW PRAM*. All our algorithms use $O(m+n)$ processors.

2 Algorithm for $2\Delta - 1$ Coloring

We start with the simple case of coloring the edges of the given graph with $2\Delta - 1$ colors. This can easily be achieved in *NC* by using the $\Delta + 1$ vertex coloring of Luby [18] on the line graph of G. This algorithm runs in $O(log^3 n \log\log n)$ time, but may use a quadratic number of processors (as the line graph of G may be quadratic in the size of G). We are seeking an efficient algorithm for this problem which uses only a linear number of processors. In the special case when Δ is bounded by a constant, the algorithm of Goldberg, Plotkin and Shannon [6] solves the problem in $O(\log^* n)$ time using $O(n)$ processors.

We use ideas from the algorithm of Lev, Pippenger and Valiant [15] for edge coloring bipartite graphs and Israeli and Shiloach's algorithm [11] for computing maximal matchings. We repeatedly "split" the graph into two graphs of smaller degree. The edge set E is divided into two parts such that the degree of the subgraph induced by each part is roughly half the degree of G. The two subgraphs are colored recursively in parallel and the colors are put together by choosing two disjoint sets of colors for the two parts.

By pairing edges incident to each vertex, it is easy to decompose a given graph into edge disjoint paths and cycles, with at most one path ending at any vertex. Paths and even length cycles split into two parts easily. The edges of each cycle (path) are colored with two colors (say 0 and 1), alternating between the two colors on some traversal of the cycle (path). The edges with color 0 form one of the subgraphs and the edges with color 1 form the other one. As there are no odd length cycles in bipartite graphs, the splitting procedure of [15] is quite simple. In the maximal matching algorithm given in [11], an Eulerian cycle is used to split the graph. But computing an Eulerian cycle seems to requires $\Theta(\log^2 n)$ time on a *CREW PRAM* when using a linear number of processors [2]. Also computing Eulerian cycles does not seem to be necessary to partition a graph in the required way.

We use the following ideas to "handle" odd length cycles. Each odd length cycle nominates one of its incident vertices as a *special* vertex. The idea is that the cycle is divided into two parts using the previous idea except that, when the special vertex is reached the first time (from some arbitrary starting vertex), both the incoming edge and the outgoing edge are assigned the same color. The only problem may be that two cycles may nominate the same vertex as special. The key observation is that if two odd length cycles share the same special vertex, then they can be merged into a single even length cycle, by "stitching" the two cycles at that vertex. Hence cycles are paired around common special vertices and each pair of odd length cycles are merged into a single even length cycle. The remaining odd length cycles do not share any of the special vertices. Finally if any special vertex has odd degree, the cycle which nominated this vertex can be combined with the unique path originating at this vertex to produce a single path. After this step, all cycles and paths can be split into two parts as stated before, thus forming the two subgraphs G_0 and G_1. For odd cycles, we also make sure that the two consecutive edges (incident on a special vertex) which get the same color are assigned to G_0. It is easy to see that G_0 and G_1 have degrees at most $\lfloor \Delta/2 \rfloor + 1$ and $\lceil \Delta/2 \rceil$ respectively.

The two subgraphs G_0 and G_1 are then colored recursively using a disjoint set of colors. Observe that we might have used as many as 2Δ colors (an extra color may have been used). In this case, we pick one of the colors and recolor the edges of the matching corresponding to this color. We observe that when $2\Delta - 1$ colors are used, every edge in the matching given by the extra color has some color "vacant" at both the incident vertices and therefore no new colors are used in this stage. Algorithm 1 implements the above ideas.

3 Implementation Details

It is obvious that the algorithm works correctly. Therefore we just discuss the important implementation details. Some of the steps which are not mentioned can be performed easily using classical algorithms like prefix sum, list ranking, etc. (see e.g. [1]). In steps 3 and 6 of the algorithm, we need to count the length of each cycle. Before this can be accomplished, the edges of the cycle (path) have to be oriented in the same direction. This can be done in $O(\log n)$ time with a linear number of processors and is discussed in detail in [2], [15]. Once this is accomplished, counting the length of the cycle and coloring the edges of the cycle with two alternating colors are simple tasks.

Step 5a can be performed in $O(\log \Delta)$ time as follows. Each odd cycle nominates one of the edges on the cycle, incident to its special vertex, to represent that cycle. Then perform a prefix sum operation using the processors of edges incident to each special vertex, and count those edges incident on that vertex which are representatives of odd cycles. Subsequently cycles can be grouped into pairs easily (by pairing edges with $2i - 1$ and $2i$ as their respective prefix sums).

Lemma 3.1 *Let G be an arbitrary graph with n vertices and m edges. Let Δ be the degree of G. The graphs G_0 and G_1 produced in Step 7 of the Algorithm 1 have degrees at most $\lfloor \Delta/2 \rfloor + 1$ and $\lceil \Delta/2 \rceil$ respectively. Hence the depth of recursion is $O(\log \Delta)$.*

Proof. Each vertex groups its incident edges into pairs. Therefore coloring the edges alternately with 0 and 1 ensures that the degree is split evenly. Only odd length cycles are a problem and are handled by the use of special vertices. Every vertex is special to at most one odd cycle. Also two consecutive edges on an odd cycle, incident to its special vertex are colored with color 0 and given to G_0. Hence the degrees of G_0

```
Input: A graph $G$. Let its degree be $\Delta$.
Output: An edge coloring of $G$ using at most $2\Delta - 1$ colors.

 1. If $\Delta \leq 2$ then, color the edges of $G$ with at most 3 colors and return.
 2. Each vertex pairs the incident edges arbitrarily into groups of two, partitioning the edges into cycles and paths.
 3. Classify the cycles into even and odd length.
 4. Each odd cycle nominates one of its vertices arbitrarily as special.
 5. For each special vertex $v$ do in parallel
     (a) Group cycles which nominated $v$ as special into pairs leaving at most one.
     (b) Each pair of cycles is stitched at $v$ producing an even length cycle.
     (c) If an odd length cycle remains with $v$ as special vertex and $deg(v)$ is odd then merge this odd cycle and the unique path from $v$ producing a single path.
 6. Color the edges of each cycle and path with colors 0 and 1 alternately. For odd cycles, two consecutive edges incident on the special vertex get color 0.
 7. Split the graph into two using the two colors, producing graphs $G_0$ and $G_1$. Let their degrees be $\Delta_0$ and $\Delta_1$ respectively.
 8. Recursively color $G_0$ and $G_1$ in parallel using $2\Delta_0 - 1$ and $2\Delta_1 - 1$ colors respectively.
 9. Color the edges of $G$ with the colors of $G_0$ and $G_1$, using disjoint colors for the two subgraphs.
10. If an extra color is used, recolor the matching corresponding to a color with least number of edges.
```

Algorithm 1: Edge coloring G with $2\Delta - 1$ colors

and G_1 are as stated in the lemma. The depth of recursion is $O(\log \Delta)$ because $\Delta - 2$ reduces at least by a factor of 2 in each splitting step. □

Theorem 3.2 *Let G be a graph as in Lemma 3.1. Algorithm 1 computes an edge coloring for G using at most $2\Delta - 1$ colors in $O(\log n \log \Delta)$ time with $O(m+n)$ processors.*

Proof. The implementation details were discussed earlier. Each splitting procedure can be implemented in $O(\log n)$ time using a linear number of processors. Also by Lemma 3.1, the depth of recursion is $O(\log \Delta)$. When the degree becomes at most 2, the graph can easily be colored with 3 colors in $O(\log^* n)$ time [6]. Therefore the algorithm runs in $O(\log n \log \Delta)$ time and uses a linear number of processors. □

4 Reducing the Number of Colors Used

We now show how the above algorithm can be modified to produce a $c\Delta$ coloring for any arbitrary constant $c > 1$. The basic idea is to use the same splitting procedure as above. Instead of going down to graphs of degree 2, we stop the recursion at an earlier stage when the degree becomes at most d (constant to be determined later, depending on c). These graphs are colored with $d+1$ colors using the procedure of Karloff and Shmoys [12]. Also the recoloring step in line 10 of Algorithm 1 is not performed.

Let $N(\Delta)$ be the number of colors used by this algorithm on a graph of degree Δ. Then we get the following recurrence.

$$\begin{aligned} N(\Delta) &\leq N(\lfloor \Delta/2 \rfloor + 1) + N(\lceil \Delta/2 \rceil), \text{ for } \Delta > d \qquad (1) \\ N(\Delta) &= \Delta + 1, \text{ for } \Delta \leq d \end{aligned}$$

We need to find a value for d such that $N(\Delta) \leq c\Delta$ for large enough values of Δ. It can be verified that the following is a possible solution for $\Delta \geq d$:

$$N(\Delta) \leq c\Delta - c \text{ and } d \geq \frac{c+1}{c-1} \qquad (2)$$

Theorem 4.1 *Let G be a graph with n vertices and m edges. Let Δ be the degree of G. It is possible to compute an edge coloring for G using at most $c\Delta$ colors for any constant $c > 1$ in $O(\log^2 n)$ time with $O(m+n)$ processors.*

Proof. As before, each splitting step takes $O(\log n)$ time and the depth of recursion is $O(\log \Delta)$. At the bottom of recursion, we need to color graphs of constant degree d

with $d+1$ colors. As mentioned before, this can be done in $O(\log^2 n)$ time using a linear number of processors. Combining all this, we get the desired result. $\square$

5 A Super Fast Coloring Algorithm

In this section, we show that a Δ^2 coloring can be computed in $O(\log^* n)$ time on a PRAM. We wish to note that such a coloring is nontrivial in the given amount of time when Δ is large but bounded by $\sqrt{n}$. The input graph has to be given in a special way (otherwise preprocessing will take $O(\log n)$ time). But such a type of input for graphs has been used before [15].

The algorithm deviates from previous algorithms significantly in the following respect. Most parallel algorithms for the edge-coloring problem use the idea of reducing the degree of the graph by a factor of two and then solving the problem recursively. Such an algorithm would take $\Theta(\log \Delta)$ phases to reach the base case and hence does not lead to an $O(\log^* n)$ time algorithm. Linial [16] has shown that there is an algorithm for computing a Δ^2 coloring in $O(\log^* n)$ time. But his algorithm is based on the existence of certain sets, and it is not known how to construct these sets. In essence he shows that such an algorithm exists and not how to find it. Our algorithm is uniform, much simpler and based entirely on graph theoretic techniques.

The algorithm works as follows. We use Δ^2 colors in this algorithm and we treat each color as an ordered pair, with each entry being in the range 1 to Δ. Each edge receives two labels between 1 and Δ, one from each vertex incident on it. Each vertex gives unique labels to all incident edges. Edges order these two labels in an arbitrary way and use these pairs as their edge colors. This procedure is sufficient for bipartite graphs to give a valid edge-coloring, provided the tuple uses the obvious order created by the bipartition instead of an arbitrary order. For general graphs, we may not have a valid coloring at this point.

It is easy to see that the number of edges of the same color being incident at a given vertex is bounded by two. Therefore the subgraph corresponding to each color forms paths and cycles. Additionally by just using local information, each edge in the graph can find its neighbors in its cycle or path in constant time (with suitable data structures). The vertices on each of these paths and cycles resolve the color conflicts as follows. We color the vertices of each path and cycle using three colors in $O(\log^* n)$ time using the algorithm of Goldberg, Plotkin and Shannon [6]. The vertex coloring partitions the vertices of each path or cycle into three independent sets. The conflicts in each path or cycle are resolved by these vertices in three steps (corresponding to the three colors). During its turn, if a vertex finds that the two edges incident on it have the same color $\langle i, j \rangle$, it swaps the two labels that the vertex initially gave these two edges, and thereby changes their colors to $\langle i, i \rangle$ and $\langle j, j \rangle$. Note that it is impossible for colors of the form $\langle i, i \rangle$ to have conflicts. Algorithm 2 is based on the above ideas.

Input: A graph G. Let its degree be Δ.
Output: An edge coloring of G using at most Δ^2 colors.

1. Each vertex assigns labels $1, 2, \cdots, \Delta$ to its incident edges arbitrarily, so that each edge gets a unique label.
2. Each edge orders the two labels that it receives arbitrarily and uses the 2-tuple as its color.
3. Consider the paths and cycles generated by each color in parallel. Color the vertices of each path with 3 colors (0, 1, and 2, say).
4. Resolve the color conflict in each path or cycle in 3 steps. In step k, the colors of edges around vertices colored k are resolved. On a path labeled $\langle i, j \rangle$, a vertex must have assigned the labels i and j to its two incident edges repectively. During its turn, if the vertex detects a color conflict, it swaps the labels to these two edges, changing their labels to $\langle j, j \rangle$ and $\langle i, i \rangle$ respectively.
5. The labels of the edges now correspond to a valid Δ^2 coloring.

Algorithm 2: Edge coloring G with Δ^2 colors

By building suitable data structures in each step, we can implement the algorithm to run in $O(\log^* n)$ time. This algorithm demonstrates that faster approximation algorithms are possible with newer ideas. We conjecture that it is possible to compute a $c\Delta$ coloring for any $c > 1$ in $O(\log n)$ time.

6 Application to Maximal Matching

The edge coloring problem is a generalization of graph matching. The maximum matching problem has been widely studied and is known to be in *RNC* [13], [19]. This problem has not been shown to be in deterministic *NC* so far. In many circumstances, it may be possible to use a "large enough" matching in the place of a maximum matching. A maximal matching serves this purpose as any maximal matching has at least half the size of a maximum matching. The maximal matching problem is easily shown to be in *NC* as a consequence of Luby's parallel algorithm for the Maximal Independent set problem [17]. But a significantly more efficient algorithm was given by Israeli and Shiloach [11]. Their algorithm runs in $O(\log^3 n \log \Delta)$ time with a linear number of processors on the *CREW PRAM* (actually their algorithm runs in $O(\log^2 n \log \Delta)$ time on the *CRCW* machine and the standard simulation of a concurrent write machine on an exclusive write machine adds an extra factor of $\log n$). If randomization is allowed, even more efficient solutions are possible. Israeli and Itai [10] demonstrate a randomized parallel algorithm for maximal matching, which runs in $O(\log n)$ time with $O(|E|)$ processors on a *CRCW PRAM*.

In the maximal matching algorithm of [11], one of the basic steps used is the algorithm `Degree-split`, which splits the edges of a graph of degree Δ into two graphs of degree $\lceil \Delta/2 \rceil$. This is achieved by using an Eulerian circuit of an augmented graph. Computation of an Eulerian tour of a graph seems to require $O(\log n)$ time on a *CRCW* machine. Our splitting step can replace their Degree-split algorithm.

Theorem 6.1 *Let G be an arbitrary graph with n vertices and m edges. Let Δ be the degree of G. It is possible to compute a maximal matching of G in $O(\log^2 n \log \Delta)$ time with $O(m+n)$ processors on a CREW PRAM.* □

References

[1] S. G. Akl. *The Design and Analysis of Parallel Algorithms.* Prentice-Hall, Englewood Cliffs, N.J., 1989.

[2] B. Awerbuch, A. Israeli, and Y. Shiloach. Finding Euler Circuits in Logarithmic Parallel Time. In *Advances in Computing Research 4: Parallel and Distributed Algorithms, Preparata, F. P., ed., JAI Press, Greenwich, CT*, pages 69–78, 1987.

[3] R. Cole and J. Hopcroft. On Edge Coloring Bipartite Graphs. *SIAM J. Comput.*, 11:540–546, 1982.

[4] M. Fürer and B. Raghavachari. An Efficient *NC* Algorithm for Finding Hamiltonian Cycles in Dense Directed Graphs. In *Proceedings of 18th International Colloquium on Automata, Languages and Programming*, pages 429–440, 1991.

[5] H. N. Gabow and O. Kariv. Algorithms for Edge Coloring Bipartite Graphs and Multigraphs. *SIAM J. Comput.*, 11:117–129, 1982.

[6] A. V. Goldberg, A. Plotkin, and G. E. Shannon. Parallel Symmetry Breaking in Sparse Graphs. *SIAM J. Disc. Math.*, 1:434–446, 1988.

[7] T. Gonzalez and S. Sahni. Open Shop Scheduling to Minimize Finish Time. *J. Assoc. Comput. Mach.*, 23:665–679, 1976.

[8] C. C. Gotlieb. The Construction of Class-teacher Time-tables. In *Proc. IFIP Congress 62, Munich, North-Holland*, pages 73–77, 1963.

[9] I. Holyer. The NP-completeness of Edge-coloring. *SIAM J. Comput.*, 10:718–720, 1981.

[10] A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Inf. Proc. Lett.*, 22:77–80, 1986.

[11] A. Israeli and Y. Shiloach. An Improved Parallel Algorithm for Maximal Matching. *Inf. Proc. Lett.*, 22:57–60, 1986.

[12] H. J. Karloff and D. B. Shmoys. Efficient Parallel Algorithms for Edge Coloring Problems. *J. Algorithms*, 8:39–52, 1987.

[13] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a Maximum Matching is in Random *NC*. *Combinatorica*, 6:35–48, 1986.

[14] E. L. Lawler and J. Labetoulle. On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming. *J. Assoc. Comput. Mach.*, 25:612–619, 1978.

[15] G. F. Lev, N. Pippenger, and L. G. Valiant. A Fast Parallel Algorithm for Routing in Permutation Networks. *IEEE Transactions on Computers*, C-30:93–100, 1981.

[16] N. Linial. Distributive Graph Algorithms - Global Solutions from Local Data. In *28th FOCS*, pages 331–335, 1987.

[17] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. Comput.*, 15:1036–1053, 1986.

[18] M. Luby. Removing Randomness in Parallel Computation Without a Processor Penalty. In *29th FOCS*, pages 162–173, 1988.

[19] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is As Easy As Matrix Inversion. *Combinatorica*, 7:105–113, 1987.

[20] V. G. Vizing. On an Estimate of the Chromatic Class of a P-graph (russian). *Diskret. Anal.*, 3:25–30, 1964.

Parallel Multiselection and Optimal Merging

Narsingh Deo Amit Jain Muralidhar Medidi

Department of Computer Science, University of Central Florida, Orlando, FL 32816

Abstract

Given two ordered multisets A and B, of sizes m and n respectively, where $m \leq n$, and a sequence of r integers $1 \leq K_1 < K_2 < \ldots < K_r \leq (m+n)$, the *multiselection* problem is to find all the K_ith, $1 \leq i \leq r$, smallest elements in $A + B$, the combined elements of the two arrays, without explicitly merging the arrays. We present an $O(\log m + \log r)$ time algorithm using r processors on the EREW PRAM. Multiselection leads to a simple and optimal merging algorithm on the EREW PRAM which runs in $O((m+n)/p + \log m + \log p))$ time with p processors, where $1 \leq p \leq (m+n)/\log(m+n)$. Our merging algorithm has only $O(p)$ data copying which compares favorably with other merging algorithms for the EREW PRAM. The distinguishing feature of our technique is selecting the correct elements to partition the merging problem, instead of partitioning first and then determining the relative positions.

1 Introduction

The problem of selecting the jth smallest element out of n unordered elements has received much attention in sequential and parallel computation because of its theoretical significance as well its application in merging, sorting and searching. Sequentially the problem can be solved in $O(n)$ time [4]. Various parallel algorithms have also been proposed for selection [1, 5, 15]. A generalization of this problem is multiselection. Here, given an unordered set of n elements and a sequence of r integers $1 \leq K_1 < K_2 < \ldots < K_r \leq n$, the problem is to find all the K_ith, $1 \leq i \leq r$, smallest elements. An efficient parallel algorithm for multiselection in an unordered set, based upon Cole's parallel selection algorithm [5], has been presented by Olariu and Wen [11].

Given two ordered multisets A and B of sizes m and n, another type of selection problem is to find the jth smallest element in $A + B$, the combined elements of the two arrays, without explicitly merging the arrays. This selection can be performed in $O(\log(min\{j, m, n\}))$ time sequentially [7, 8]. Finding the median of $A + B$ is a special case of this problem, and has been discussed in [2, 14]. Selection in two ordered sets, and in particular, median-finding is used in parallel merging and sorting [2, 7, 14].

It is natural to inquire if multiselection in two ordered multisets can be performed optimally or even efficiently in parallel. An optimal solution for multiselection leads to an elegant parallel merging algorithm which is optimal in time and cost. Parallel merging algorithms in [2, 7] employ either sequential median or selection algorithm. Consequently, the resulting parallel algorithms do not achieve optimal time-complexity, even though they are cost-optimal. Parallel algorithms for merging in [3, 9, 13, 14] use various techniques to overcome the difficulty of multiselection. Since merging is a fundamental non-numeric operation which frequently arises in such diverse applications as computational geometry and relational database systems, there is a need for simpler parallel algorithms for merging that are fast and easier to implement.

First, we present a new sequential algorithm for selection in two sorted arrays and then use it to develop a parallel algorithm for multiselection. Our algorithm uses r processors, on the exclusive read, exclusive write parallel random access machine (EREW PRAM, see [10] for a definition), to perform multiselection for the set of queries $\{K_1, \ldots, K_r\}$ in $O(\log m + \log r)$ time, where we assume that $m \leq n$. This multiselection algorithm leads to a simple merging algorithm which runs in $O((m+n)/p + \log m + \log p))$ time with p processors, where $1 \leq p \leq (m+n)/\log(m+n)$. In the rest of the paper we will use the term selection, for convenience, to mean selection in two sorted arrays.

2 Selection in Two Sorted Arrays

First we present a different view of the selection algorithm, such that finding the jth smallest element can be reduced to selecting the median of the merged appropriate subarrays of A and B without explicitly merging them. Without loss of generality we assume $m \leq n$. Consider the case of selecting the jth element, $1 \leq j \leq m$. Since

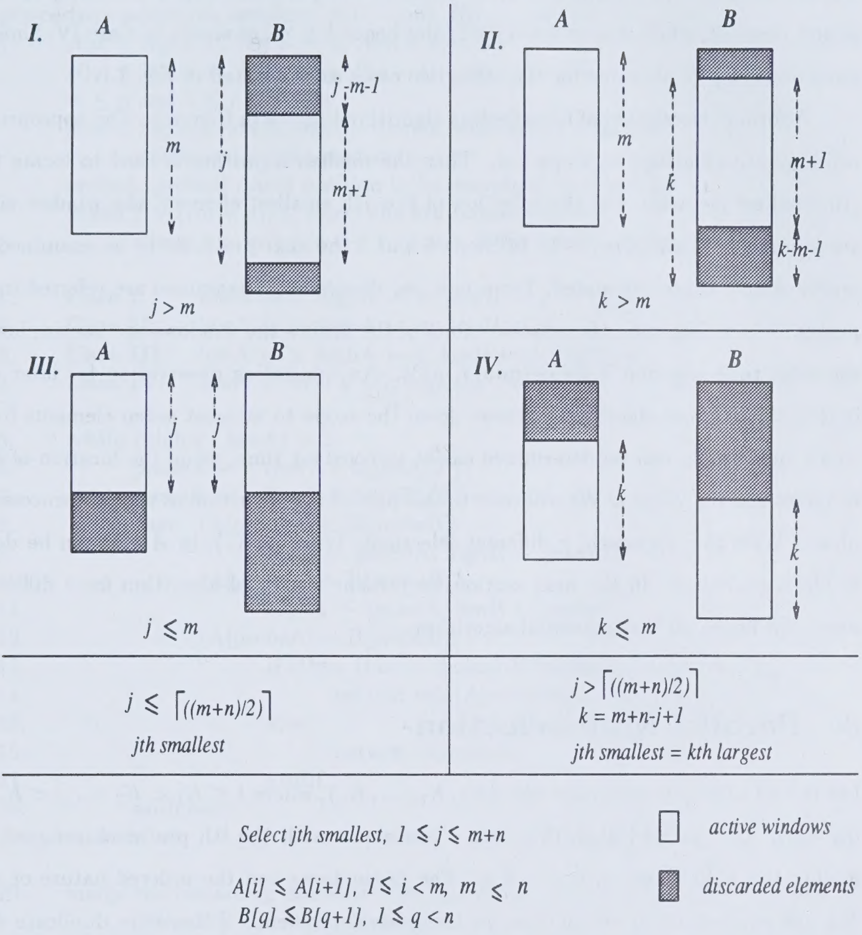

I.
A
B
m
j
j-m-1
m+1
j > m
II.
A
B
m
k
m+1
k-m-1
k > m
III.
A
B
j
j
j ⩽ m
IV.
A
B
k
k
k ⩽ m
j ⩽ ⌈((m+n)/2)⌉
jth smallest
j > ⌈((m+n)/2)⌉
k = m+n-j+1
jth smallest = kth largest
Select jth smallest, 1 ⩽ j ⩽ m+n
A[i] ⩽ A[i+1], 1 ⩽ i < m, m ⩽ n
B[q] ⩽ B[q+1], 1 ⩽ q < n
active windows
discarded elements

Figure 1: Reduction of selection to median finding

$j \leq m \leq n$, the required element can only be in the subarrays $A[1..j]$ and $B[1..j]$. Thus the median of the $2j$ elements in these subarrays is the jth smallest element. This reduction is depicted as Case III in Fig. 1. The appropriate subarray is referred to as a *window* from now on. When $j > n$, we can view the problem as that of finding the kth largest element, where $k = m + n - j + 1$, and hence $k \leq m$, as shown in Case IV. Similar reductions can be obtained for the other two cases as illustrated in Fig. 1.

A formal description of the selection algorithm is given in Figure 2. The appropriate windows are identified in Steps 1–4. Then the median algorithm is used to locate the jth smallest element. For the selection of the jth smallest element, the window sizes are bounded by $\min\{j, m\} + 1$. In Steps 6 and 7 the next positions to be examined in arrays A and B are calculated. From now on, the elements examined are referred to as *probes.* Depending upon the relative values of the probes the windows are halved; hence the total time required is $O(\log(\min\{j, m\}))$. An interesting observation for later use is that this median algorithm narrows down the scope to at most seven elements from which the median can be determined easily, in constant time, using the location of the windows and the value j. We will refer to this part of the algorithm as the postprocessing phase. With this approach, r different selections, $\{K_1, \ldots, K_r\}$, in $A + B$ can be done in $O(r \log m)$ time. In the next section we present a parallel algorithm for r different selections based on the sequential algorithm.

3 Parallel Multiselection

Let the set of selection positions be $\{K_1, K_2, \ldots, K_r\}$, where $1 \leq K_1 < K_2 < \ldots < K_r \leq (m + n)$. Our parallel algorithm uses r processors with the ith processor assigned to finding the K_ith element, $1 \leq i \leq r$. The distinctness and the ordered nature of the K_is are not significant restrictions on the general problem. If there are duplicate K_is or if the selection positions are unsorted, both can be remedied in $O(\log r)$ time using r processors [6]. On a CREW PRAM the problem admits a trivial solution, as each processor can carry out the selection independently. However, on an EREW PRAM the problem becomes interesting because read conflicts have to be avoided.

Let us first consider only those K_is that fall in the range $[m + 1 \ldots \lceil (m + n)/2 \rceil]$,

```
procedure select_jth_smallest( $A[1\ldots,m]$, $B[1,\ldots,n]$, $j$)
{       $A[p] \le A[p+1]$, $1 \le p \le m$, $A[m+1] = \infty$
        $B[q] \le B[q+1]$, $1 \le q \le n$, $B[n+1] = \infty$
        $m \le n$ and $1 \le j \le m+n$
        [lowA, highA], [lowB, highB]: current windows in $A$ and $B$
                which contain the median
        probeA, probeB : next position to be examined in $A$ and $B$
        When $j > \lceil (m+n)/2 \rceil$ select the $k$th largest element,
                where $k = m+n-j+1$, in $A+B$ for better efficiency }

        Case I:     lowA $\leftarrow$ 1; highA $\leftarrow$ $m$; lowB $\leftarrow$ $j-m$; highB $\leftarrow$ $j$.
        Case II:    lowA $\leftarrow$ 1; highA $\leftarrow$ $m$; lowB $\leftarrow$ $n-k+1$; highB $\leftarrow$ $j$.
        Case III:   lowA $\leftarrow$ 1; highA $\leftarrow$ $j$; lowB $\leftarrow$ 1; highB $\leftarrow$ $j$.
        Case IV:    lowA $\leftarrow$ $m-k+1$; highA $\leftarrow$ $m$; lowB $\leftarrow$ $n-k+1$; highB $\leftarrow$ $n$.

        while (highA - lowA) $> 2$
                probeA $\leftarrow$ $\lfloor$ (lowA+ highA)/2 $\rfloor$
                probeB $\leftarrow$ $\lfloor$ (lowB + highB)/2 $\rfloor$
                case  (A[probeA] < B[probeB]) :
                            lowA $\leftarrow$ probeA; highB $\leftarrow$ probeB
                      (A[probeA] > B[probeB]) :
                            highA $\leftarrow$ probeA; lowB $\leftarrow$ probeB
                      (A[probeA] = B[probeB]) :
                            if (Case II or IV holds) & (probeA + probeB < j)
                                    return min(A[probeA+1], B[probeB+1])
                            else
                                    return A[probeA]
                            endif
                endcase
        endwhile

        merge the remaining elements from $A+B$ (at most 7) and
                determine the required element.
endprocedure
```

Figure 2: **Algorithm for selecting the jth smallest element in $A+B$**

that is, those for which Case I, in Figure 1, holds. All of these selections initially share the same probe in array A. Let $m < K_l < K_{l+1} < \ldots < K_h \leq \lceil (m+n)/2 \rceil$ be a sequence of K_is that share the same probe in A at any time. Note that these selections will have different probes in array B. We refer to such a sequence of selections as a *chain*, following the terminology of Paul *et al* [12]. Let the probe in array A be x for this chain, and the probes in array B be $y_l \leq y_{l+1} \leq \ldots \leq y_h$. The processor associated with K_lth selection is active for the chain. This processor compares x with y_l and y_h, and based on these comparisons the following actions take place:

- $x < y_l$: The chain stays intact.
- $x = y_l$:
    - ⋆ $x = y_h$: The chain becomes inactive.
    - ⋆ $x < y_h$: The chain is split into two chains.
- $x > y_l$
    - ⋆ $x \leq y_h$: The chain is split into two chains.
    - ⋆ $x > y_h$: The chain stays intact.

Note that three comparisons are required to determine if the chain stays intact or has to be split. When the chain stays intact, the window in array A remains common for the whole chain. The processor p_l computes the size of the new common window in array A. The new windows in the array B can be different for the selections in the chain, but they all shrink by the same amount, and hence the offset from the initial window is same for all the selections. However, the windows may shrink up or down, hence two offsets are sufficient for the entire chain. Three comparisons made by the processor handling the chain determine the windows for all the selections in the chain (when the chain stays intact). When the chain becomes inactive, it is within 7 elements to compute the required median for all the selections in the chain. The chain does not participate in the algorithm any more except for the postprocessing phase.

When a chain splits, processor p_l activates the processor $p_{\lceil (l+h)/2 \rceil}$ to handle the chain $K_{\lceil (l+h)/2 \rceil}, \ldots, K_h$. It remains in charge of the chain $K_l, \ldots, K_{\lceil (l+h)/2 \rceil - 1}$. It passes the current probe position and value in the array A, the offsets for the array B, and the parameter h. In the same stage both these processors again check if their respective chains remain intact. If the chains remain intact they move on to a new probe position. Otherwise they stay at the current probe position to be processed in the next stage. Similar to the proof given in Paul et al [12], it can be shown that at most two chains remain at a probe position after this stage. Also there can be at most two new chains arriving at a probe position from previous stages. All of this processing within a stage can be performed in $O(1)$ time as at most four processors may read a probe. When a chain splits into two chains, the windows of these chains in A will overlap only at the probe that split them. Any possible read conflicts at this common element can happen only during the postprocessing phase (which can be handled as described in the next paragraph). Hence all of the processing can be done without any read conflicts.

At each stage the chains are either splitting into two halves or the window size is halved. Hence after at most $O(\log m + \log r)$ stages each selection process must be within seven elements of the required position. At this point, each processor has a window of size at most 3 in A and 4 in B. If the windows have any elements in common, the values can be broadcasted in $O(\log r)$ time, such that each processor can then carry out the required postprocessing in $O(1)$ time.

However, the inactive chains may contain at most r positions, due to repeated elements. But the processor handling the chain knows the required element for all the selections in the inactive chain except for the last one. Broadcasting the required element needs only $O(\log r)$ time. Note that the total amount of data copied is only $O(r)$.

Of the rest, those K_is falling in Case II (of Fig. 1) can be handled in exactly the same way as the ones in Case I. The chaining concept can be used if $O(1)$ comparisons can determine the processing for the whole chain. In Cases III and IV, different selections have windows of different sizes in both the arrays. Hence, chaining cannot be directly used as in Cases I and II. However, we can reduce Case III (IV) to I (II). To accomplish this reduction, imagine array B to be padded by m elements with value $-\infty$ in locations $-m+1$ to 0 and by m elements with value ∞ in locations $n+1$ to $n+m$. Let this array be denoted as C (which need not be explicitly constructed). Selecting the $1 \leq j \leq m+n$ element in $A[1..m]$ and $B[1..n]$ is equivalent to selecting the $m+1 \leq (j+m) \leq 2m+n$ element in the arrays $A[1..m]$ and $C[1..2m+n]$. Thus, selections in Case III (IV) in the arrays A and B become selections in Case I (II) in the arrays A and C.

For clarity of presentation we have increased the window sizes in case III (IV) to m in array A and $m+1$ in C. For efficiency, it is better to truncate array A and B depending upon the maximum (minimum) value of a selection in case III (IV) before using the reduction. If all selection positions fall either in interval [1 ... m] (Case III) or [n+1 ... n+m] (Case IV) then r selections can be performed by r processors in time

$$O(\log(\max\{\max_{K_i \in [1\ldots m]} K_i,\ m+1-\min_{K_j-n \in [1\ldots m]} K_j - n\})).$$

Any selection in the interval $[m \ldots n]$ dominates the time complexity. In such a case, we note that all of the selections in different cases can be handled by one chain with the appropriate reductions. Hence we have the following:

Theorem 3.1 *Given r selection positions $\{K_1, \ldots, K_r\}$, all of the selections can be made in $O(\log m + \log r)$ time using r processors.*

4 Parallel Merging

Now, consider the problem of merging two sorted sequences A and B of length m and n, respectively. Hagerup and Rub [9] have presented an optimal algorithm which runs in $O(\log(m+n))$ time using $((m+n)/\log(m+n))$ processors on an EREW PRAM. The algorithm recursively calls itself once and then uses Batcher's method for merging two sub-sequences for partitioning. Also in order to avoid read conflicts, parts of the sequences are copied by some processors.

Akl and Santoro [2] and Deo and Sarkar [7] have used selection as a building block in parallel merging algorithms. Even though these algorithms are cost-optimal, their time complexity is $O(\log^2(m+n))$ on the EREW PRAM. By solving the parallel multiselection problem, we are able to obtain a simpler cost-optimal merging algorithm with time complexity $O(\log(m+n))$ requiring an $((m+n)/\log(m+n))$ processor EREW PRAM. The algorithm can be expressed as follows:

1. Find the $i\lfloor\log(m+n)\rfloor$, $i = 1, 2, \ldots, j (= \frac{(m+n)}{\lfloor\log(m+n)\rfloor})$, ranked element using multiselection. Let the output be two arrays $R_A[1..j]$ and $R_B[1..j]$, where $R_A[i] \neq 0$

```
    implies that $A[R_A[i]]$ is the $(i\lfloor\log(m+n)\rfloor)$th element and $R_B[i] \neq 0$ implies that
    $B[R_B[i]]$ is the $(i\lfloor\log(m+n)\rfloor)$th element.

2.  Let $R_A[0] = R_B[0] = 0$
    for $i = 1, \ldots, j$ do
      if $R_A[i] = 0$ then $R_A[i] = i * \lfloor\log(m+n)\rfloor - R_B[i]$
      else $R_B[i] = i * \lfloor\log(m+n)\rfloor - R_A[i]$

3.  for $i = 1, \ldots, j$ do
      merge $(A[R_A[i-1]+1]..A[R_A[i]])$ with $(B[R_B[i-1]+1]..B[R_B[i]])$.
```

Steps 1 and 3 both take $O(\log(m+n))$ time using $(m+n)/\log(m+n)$ processors. Step 2 takes $O(1)$ time using $(m+n)/\log(m+n)$ processors. Thus the entire algorithm takes $O(\log(m+n))$ time using $(m+n)/\log(m+n)$ processors, which is optimal. The total amount of data copied in Step 1 is $O((m+n)/\log(m+n))$, since $r = (m+n)/\log(m+n)$, and compares favorably with $O(m+n)$ data copying required by Hagerup and Rub's [9] merging algorithm.

If fewer processors, say p, are available, the proposed parallel merging algorithm can be adapted to perform p multiselections and will require a total of $O((m+n)/p + \log m + \log p))$ time.

Concluding Remarks: We have presented a parallel merging algorithm for the EREW PRAM which requires less data copying. In comparison to the merging algorithms presented by Bilardi and Nicolau [3] and Hagerup and Rub [9], the processor allocation in our algorithm is simpler. The distinguishing feature of our technique is selecting the correct elements to partition the problem, instead of partitioning first and then determining the relative positions. Indeed, we solve a more general problem of multiselection in two sorted arrays which could be applied in other problems similar to merging.

Clearly, the merging algorithm presented leads to another cost-optimal $O(\log^2 n)$-time sorting algorithm for the EREW PRAM. We are currently investigating worst-case number of comparisons required in our multiselection and merging algorithms.

References

[1] S. G. Akl. An optimal algorithm for parallel selection. *Information Processing Letters*, 19(1):47–50, July 1984.

[2] S. G. Akl and N. Santoro. Optimal parallel merging and sorting without memory conflicts. *IEEE Transactions on Computers*, C-36(11):1367–1369, November 1987.

[3] G. Bilardi and A. Nicolau. Adaptive bitonic sorting: An optimal parallel algorithm for shared memory machines. *SIAM Journal on Computing*, 18(2):216–228, April 1989.

[4] M. Blum, R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973.

[5] R. J. Cole. An optimally efficient selection algorithm. *Information Processing Letters*, 26:295–299, January 1988.

[6] R. J. Cole. Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, August 1988.

[7] N. Deo and D. Sarkar. Parallel algorithms for merging and sorting. *Information Sciences*, 51(121–131), 1990. Preliminary version appeared in *Proceedings of the Third International Conference on Supercomputing*, May (1988), pages 513–521.

[8] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X+Y$ and matrices with sorted columns. *Journal of Computer and System Sciences*, 24:197–208, 1982.

[9] T. Hagerup and C. Rub. Optimal merging and sorting on the EREW PRAM. *Information Processing Letters*, 33:181–185, December 1989.

[10] R. Karp and V. Ramachandran. Parallel algorithms for shared memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, Cambridge, MA, 1990. The MIT Press.

[11] S. Olariu and Z. Wen. An efficient parallel algorithm for multiselection. *Parallel Computing*, 17:689–693, 1991.

[12] W. Paul, U. Vishkin, and H. Wagener. Parallel dictionaries on 2-3 trees. In *Proceedings of ICALP, 154*, pages 597–609, July 1983.

[13] Y. Shiloach and U. Vishkin. Finding the maximum, merging, and sorting in a parallel computation model. *Journal of Algorithms*, 2:88–102, 1981.

[14] P. J. Varman, B. R. Iyer, B. J. Haderle, and S. M. Dunn. Parallel merging: Algorithm and implementation results. *Parallel Computing*, 15:165–177, 1990.

[15] U. Vishkin. An optimal parallel algorithm for selection. In *Advances in Computing Research*, volume 4, pages 79–86, 1987.

Generating Binary Trees in Parallel[1]

Selim G. Akl[2] and Ivan Stojmenovic[3]

[2]Dept. of Computing and Inf. Science, Queen's Univ., Kingston, Ont. K7L 3N6 Canada

[3]Computer Science Department, University of Ottawa, Ottawa, Ont. K1N 9B4, Canada

Abstract

We present two cost-optimal parallel algorithms for generating binary trees. Using a known inversion table representation, and a new breadth first search (BFS) parent array encoding of trees as integer sequences, our first algorithm generates all tree sequences in lexicographic order. It uses a linear array of *n* processors (where *n* is the number of nodes in the tree), each having constant size memory and each being responsible for producing one node of a given tree. The algorithm can be extended to produce parent-children relations among nodes of trees, as part of the output, in constant delay per tree. If the new BFS encoding is used, then the address of the parent of each node is known directly as part of the representation. By contrast, the address of the children of a node, and access to a parent or child, can be provided trivially but on the more powerful EREW PRAM model. We also introduce a BFS representation of binary trees which allows constant time determination of a node's children on a linear array of processors. Our second parallel algorithm for generating all binary trees with *n* nodes in lexicographic order uses this representation. This algorithm can also be used to generate well formed sequences of left and right parentheses. Both BFS representations proposed in this paper lead directly to efficient sequential tree generation algorithms that enable fast inclusion of parent-children relations. Furthermore, the two new representations, as well as the sequential and parallel algorithms they yield, are generalized for the case of *t*-ary trees.

1. Introduction

The combinatorial problem of generating binary trees is concerned with generating binary trees with *n* nodes in some order. The binary trees we intend are rooted, and each node has 0, 1, or 2 children. Each child is either a *left* child or a *right* child. Many ingenious generating algorithms [10,14,18,20,26,29,32,33,34,35,36,37,38,41,43,45,48,50,51,53] for performing this task have been discovered. A list of all shapes of trees might be used to search for a counter-example to some conjecture, or to test and analyze an algorithm for its correctness or computational complexity.

In this paper we describe two parallel algorithms for generating binary trees. To our knowledge these are the first parallel algorithms for generating all $B(n) = C(2n,n)/(n+1) = (2n)!/(n!(n+1)!)$ binary trees with *n* nodes ($C(i,j)$ denotes the binomial coefficient). Both algorithms satisfy several optimality criteria. In order to justify this claim, we begin by listing some desirable properties of any algorithm that generates combinatorial objects in general, and binary trees in particular.

Property 1. The trees are listed in lexicographic order, i.e. if $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$ are representations of trees, then A precedes B lexicographically if and only if, for some $j \geq 1$, $a_i = b_i$ when $i < j$, and $a_j < b_j$.

---

[1] This research was supported by the Natural Sciences and Engineering Research Council of Canada

Property 2. The algorithm is cost-optimal, i.e. the number of processors it uses multiplied by its running time matches - up to a constant factor - a lower bound on the number of operations required to solve the problem. The time to output each tree in full is counted. Optimal sequential algorithms run in $O(n * B(n))$ time, since it takes $O(n)$ time to produce a tree.

Property 3. The time required by the algorithm between any two consecutive trees it produces is constant. A constant time delay between outputs is particularly important in applications where the output of one computation serves as input to another.

Property 4. The model of parallel computation should be as simple as possible. Arguably, the simplest such model is a linear array of m processors, indexed 1 through m, where each processor i $(2 \leq i \leq m-1)$ is connected by bidirectional links to its immediate left and right neighbors, $i-1$ and $i+1$, and processors 1 and m are each connected to one neighbor. This model is practical, as it is amenable to VLSI implementation [1].

Property 5. Each processor needs as little memory as possible, preferably a constant number of words, each of $\log n$ bits and hence capable of storing an integer no larger than n. This implies that no processor can store an array of size n, or a counter up to $n!$.

Property 6. The algorithm should produce all trees with n nodes for a given n.

Our two parallel algorithms for generating binary trees satisfy all six properties listed above and are therefore optimal in more than one sense. Parallel algorithms exist that satisfy these criteria for generating other combinatorial objects such as permutations [5], combinations [3], derangements [4], subsets and equivalence relations [46]. There are also algorithms that do not satisfy all listed properties; they include algorithms for permutations [1,2,9,24], and combinations [1,2,8,9].

A significant body of knowledge exists concerning sequential tree generation. Most papers begin by proving an equivalence between some set of integer sequences (we will call them *tree sequences*) and the set of binary trees. Once the equivalence is established, an algorithm is presented to generate all tree sequences. Similarly, our algorithms generate all tree sequences (and therefore provide the first parallel analogues to known sequential techniques).

In most references, tree sequences are generated in lexicographic order. Each of these generation algorithms causes trees to be generated in a particular order. In property 1, the lexicographic order of trees refers, more precisely, to the lexicographic order of the corresponding tree sequences. The most frequent orders are the A-order and the B-order (the latter is defined in section 2). In particular, a well-known notation for binary trees is the so-called inversion table representation [18]. In this paper we use this notation as well as two new representations based on Breadth First Search (BFS). One of our algorithms generates tree sequences in lexicographic order but lists trees in two different orders depending on whether the inversion table or BFS representation is chosen.

One property of existing sequential binary tree generation techniques is that they do not generate the usual parent-children relationships among nodes. It is clear, however, that these relationships may in some cases be of great importance given the meaning of trees as data structures. Here we extend our parallel algorithms to provide this information. We refer to tree sequences accompanied with the parent-children relations simply as binary trees. Our first parallel algorithm for generating tree sequences can be extended to provide parent-children relations in two ways, depending on the chosen representation. If the inversion table representation is used, then the algorithm can be extended in a non-trivial manner and will satisfy all listed properties except Property 4: it will require a more powerful model, namely the EREW PRAM (exclusive read exclusive write parallel random access machine), in which the processors share a common memory through which they communicate, but no two processors can gain access to the same memory location simultaneously for reading or writing [1]. Since in general there is no simple path passing through all vertices of a tree, an algorithm, running on a linear array of processors with constant delay per tree, cannot provide constant time access between parents and children; however this does not mean that the address of the parent and children of a node cannot be maintained in constant time on the same model. This is accomplished by encoding trees in a new fashion, using a parent sequence of nodes listed in BFS order. This gives a new sequential algorithm for generating all trees which includes parent-children relations by a trivial extension. It also obviously provides the address of the parent of each node in a parallel algorithm running on a linear array of processors. In addition, the address of the children of a node and access to a parent or child are provided on the EREW PRAM model in a straightforward manner.

The second BFS representation proposed in this paper allows constant time determination of a node's children on a linear array of processors. It also leads to a sequential algorithm as well as a parallel algorithm for generating all binary trees on a linear array of processors. These two algorithms can be used directly to generate all well formed sequences of left and right parentheses, by exploiting the one-to-one correspondence between the latter and binary trees. Finally, the two new representations, and the sequential and parallel algorithms derived from them are generalized for the case of *t*-ary trees, i.e. trees in which each node has at most *t* distinguished children, for a given constant *t*.

2. Binary tree representations

The binary tree representation, called the inversion table, is introduced in [18] and used in [28]. Given a binary tree *T* having *n* nodes, let the root be labeled with 0. Inductively we define a labeling for every node of *T* as follows: if a node *u* is a left child of its parent, then label *u* with 1 plus the value of the label of its parent; if *u* is a right child of its parent, label *u* with the same label as its parent. After having labeled the nodes of *T*, do a preorder traversal of *T*, writting down the labels as each node is visited, to form a sequence $x_1 \dots x_n$ representing *T*. The sequence $x_1 \dots x_n$ is a tree sequence iff $x_1=0$ and $0 \le x_j \le x_{j-1}+1$ for $2 \le j \le n$. For example, for $n=3$ the following is the list of all tree sequences with 3 nodes: 000, 001, 010, 011, 012. The tree in Fig. 1 is labeled as 001201. The parallel tree sequence algorithm described below uses this representation.

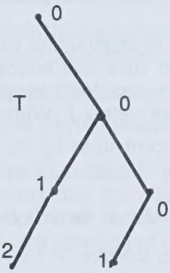


Fig. 1

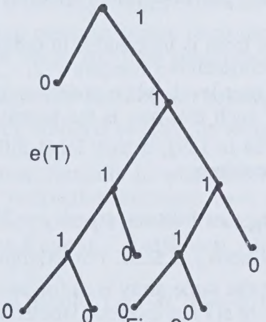


Fig. 2

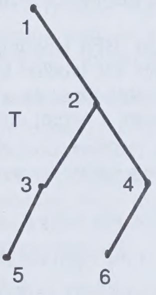


Fig. 3

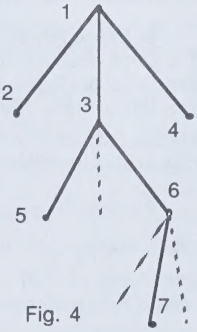


Fig. 4

A tree is regular if each node has either 0 or 2 children. In [19] a one-to-one correspondence between regular trees with $2n+1$ nodes and binary trees with *n* nodes is established by matching binary tree *T* with extended tree *e(T)*, obtained by replacing empty children of *T* with real nodes (see Fig. 1 and 2).

A related tree representation is independently introduced in [33,51] and used in [10,11]. A binary tree *T* having *n* nodes can be represented by a bitstring $b_1 b_2 \dots b_{2n}$ with *n* zeros and *n* ones that satisfies the prefix property: the number of 0's is always less than or equal to the number of 1's while scanning from b_1 to b_{2n} (i.e. $b_1+\dots+b_i \ge i/2,\ 1 \le i \le 2n$). Here each 1 corresponds to a node of *T* while 0's correspond to the leaves of *e(T)*. The bitstring encodes *e(T)*, in preorder traversal. For example, tree *T* in Fig. 1 is extended to tree *e(T)* in Fig. 2 and is coded as 101110001100 (the last 0 is omited). If the positions of 1's in a given sequence

$b_1 \ldots b_{2n}$ are marked in a separate sequence $z_1 \ldots z_n$ (called z-sequence in [51]) then, due to the prefix property, we get: $z_1=1$, $z_{j-1} < z_j \le 2j-1$ for $2 \le j \le n$. For example, 123, 124, 125, 134, 135 correspond to bitstrings 111000, 110100, 110010, 101100, and 101010, respectively; T of Fig. 1 is coded as 1,3,4,5,9,10.

Given a regular tree P, let, let r_P be the number of children of the root of P, i.e. $r_P=2$ or $r_P=0$. Let P_L and P_R denote the left and right subtrees of P. Two regular trees P and Q are in *B-order*, $P<Q$, if: 1) $r_P<r_Q$, 2) $r_P=r_Q$, and $P_L<Q_L$ or 3) $r_P=r_Q$, and $P_L=Q_L$, and $P_R<Q_R$.

Two binary trees T' and T'' are in B-order, $T'<T''$ if $e(T')<e(T'')$.

The algorithms in [41,48,51] generate binary trees in B-order. In [51,10,11] it is proved that the B-order of trees matches the lexicographic order of bitstring sequences $b_1 \ldots b_{2n}$. It is easy to verify that the lexicographic order of inversion table sequences corresponds to the B-order of binary trees. As a consequence, the inversion table and z-sequences are related by $x_i=2i-1-z_i$. Therefore the sequential algorithms for generating trees [51,33,10,11] can be used to generate all inversion table sequences, using a simple transform. Note that there are algorithms for generating trees in orders that are different from the B-order (e.g. [34,35,36,45,53]).

3. New binary tree representation and generating algorithm

We introduce a new binary tree notation in order to faciliate establishing parent-children relations in binary trees. Known sequential algorithms generate tree sequences, and the inclusion of parent-child relations requires adding a decoding procedure, "usually at a cost of greatly complicating the algorithm and/or invalidating the run time analysis" [37]. Our notation provides a simple sequential algorithm that extends trivially to add parent-children relations. We beleive that the notation introduced here leads also to the fastest sequential algorithm for generating binary trees, if the cost of including parent-children relations is taken into account.

In [39] a left-to-right BFS labeling of trees is introduced in order to generate all subtrees of a given tree. All nodes are labeled by consecutive integers $1,2,\ldots,n$ such that nodes on a lower level are labeled before those on a higher level, while nodes on the same level are labeled from left to right. [39] uses a parent array such that par_i is the parent of vertex i. While such notation suffices for the problem considered in [39], it may label different trees by the same array, and is therefore not suitable for our problem.

We introduce new parent array $p_1 \ldots p_n$ as follows: $p_1=1$, $p_i=2j$ if i is the left child of node j, and $p_i=2j+1$ if i is the right child of node j, $2 \le i \le n$. For example, the tree on Fig. 3 has parent array 134568. Another way to define the same array is as follows: a BFS labeling of both T and $e(T)$ is performed; then p_i is the label in $e(T)$ of the node labeled i in T.

Obviously, it follows that $p_{i-1} < p_i \le 2i-1$ (because $j<i$ in the above definition) for $2 \le i \le n$. Therefore sequences $p_1 \ldots p_n$ are defined in the same way as z-sequences [51]. However, the same sequence refers to different trees in these two notations (see Fig. 1 and 3 where the same tree is encoded as 134568 in the parent array notation and as 1,3,4,5,9,10 in the z-sequence notation). On the other hand, lexicographic order is the same for the parent array and z-sequence notations, and the same algorithms can be used for generating all such sequences. In particular, it means that the sequential algorithm [51] for generating z-sequences can be used to generate all tree sequences. Each node already knows its parent because of the parent array notation while children of each node can be determined indirectly by informing all parents about existing children. The algorithm below is extended from [51] to include parent-children relations.

```
for $i \leftarrow 1$ to $n$ do {$leftch_i \leftarrow 0$; $rightch_i \leftarrow 0$};
for $i \leftarrow 1$ to $n$ do $\{p_i \leftarrow i; parent_i \leftarrow i/2;$ if $i$ even then $leftch_{i/2} \leftarrow i$ else $rightch_{i/2} \leftarrow i\}$
repeat
  report binary tree; $j \leftarrow n;$
  while $p_j<2j-1$ and $j>1$ do$\{k=p_i/2;$ if $p_i$ even then $leftch_k \leftarrow 0$ else $rightch_k \leftarrow 0; j \leftarrow j-1\};$
  $p_j \leftarrow p_j+1;$ for $i \leftarrow j+1$ to n do $p_i \leftarrow p_{i-1}+1;$
  for $i \leftarrow j$ to $n$ do $\{k \leftarrow p_i/2; parent_i \leftarrow k;$ if $p_i$ even then $leftch_k \leftarrow i$ else $rightch_k \leftarrow i\}$
until $p_1=2$
```

4. Parallel generation of tree sequences

In this section we present a parallel algorithm that generates tree sequences of the form $x_1,..., x_n$ as defined in section 2. If the inversion table representation is meant, then the algorithm generates tree sequences in B-order. The same algorithm can be used to generate tree sequences in lexicographic order in the new notation introduced in section 3 (recall that the same sequence refers to different trees and thus the order of generating trees is different) without any change, except that $p_i=2i-1-x_i$, for $1 \le i \le n$, is obtained easily once x_i is generated.

Our algorithm generates the tree sequences on a linear array of n processors, indexed 1 to n, and n variables $x_1, x_2, ..., x_n$. Each processor i is responsible for maintaining x_i by reading only data from processors $i-1$ and $i+1$. Processors 1 to $n-1$ will always produce the same element unless they are advised to change their output. Processor n produces elements between 0 and $x_{n-1}+1$. This is called a *run* of processor n. Whenever $x_n=0$ processor n initiates a message that searches for processor *tp*, called the turning point, that keeps an element x_{tp} which should be increased by 1 after processor n finishes producing its current run. *tp* is the maximal index smaller than n such that $x_{tp} \le x_{tp-1}$. Each step in the search corresponds to producing a tree. After the backtracking message search finds the turning point *tp*, processors $j \ge tp$ wait for an appropriate number of steps, and then simultaneously x_{tp} increases by 1 while x_j for $j>t$ becomes 0. There is enough time for message passing because the message path is $\le x_{n-1}+1$ (it decreases staying positive), which is exactly the length of current run of processor n.

The algorithm is given formally in what follows. The processors operate synchronously; this means that the r-th instruction executed by a processor does not start until the $(r-1)$-st instruction of all the other processors has been executed. A new tree sequence is produced at the end of each iteration of the **Repeat ...until** loop. All processors will terminate simultaneously.

```
For $i \leftarrow 1$ to $n$ do in parallel { $x_i \leftarrow 0;$ $w_i \leftarrow 0;$ (*waiting time*)
  $m_i \leftarrow 0;$ (*message to search for the turning point tp *) $term_i \leftarrow false;$ (*termination flag*)
  Repeat
    if $m_i=1$ and $i<n$ and $x_i \le x_i-1$ then $m_i \leftarrow 2;$ (*turning point tp found*)
    if $w_i=1$ and $m_i=0$ then $x_i \leftarrow 0;$ (*update elements that follow tp *)
    if $w_i=1$ and $m_i=2$ then $\{x_i \leftarrow x_i+1; m_i \leftarrow 0$ }; (*increase element in tp *)
    if $m_{i+1}=1$ and $i<n$ then $\{m_i \leftarrow 1; w_i \leftarrow w_{i+1}$ } else if $m_i=1$ then $m_i \leftarrow 0;$
    if $w_i \ge 1$ then $w_i \leftarrow w_i-1;$
    if $x_n=0$ and $w_n=0$ then $\{m_n \leftarrow 1; w_n \leftarrow x_{n-1}+2$ } (*start search *) else $x_n \leftarrow x_n+1;$
    if $w_i=i$ then $term_i \leftarrow true$ ; output $x_i$
  until $term_i=true$ and $w_i=1$ }
```

The algorithm is traced for *n=4,* and the following values are obtained.

$x_1x_2x_3x_4$	$m_1m_2m_3m_4$	$w_1w_2w_3w_4$	$term_1$	$term_2$	$term_3$	$term_4$
0 0 0 0	0 0 0 1	0 0 0 2	false	false	false	false
0 0 0 1	0 0 1 0	0 0 1 1	false	false	false	false
0 0 1 0	0 0 0 1	0 0 0 3	false	false	false	false
0 0 1 1	0 0 1 0	0 0 2 2	false	false	false	false
0 0 1 2	0 1 0 0	0 1 1 1	false	false	false	false
0 1 0 0	0 0 0 1	0 0 0 2	false	false	false	false
0 1 0 1	0 0 1 0	0 0 1 1	false	false	false	false
0 1 1 0	0 0 0 1	0 0 0 3	false	false	false	false
0 1 1 1	0 0 1 0	0 0 2 2	false	false	false	false
0 1 1 2	0 0 2 0	0 0 1 1	false	false	false	false
0 1 2 0	0 0 0 1	0 0 0 4	false	false	false	true
0 1 2 1	0 0 1 0	0 0 3 3	false	false	true	true
0 1 2 2	0 1 0 0	0 2 2 2	false	true	true	true
0 1 2 3	1 0 0 0	1 1 1 1	true	true	true	true

Theorem 1. The algorithm described above generates all tree sequences with *n* nodes in lexicographic order and with constant delay per tree on a linear array of *n* processors, thus achieving an optimal cost of $O(n*B(n))$; furthermore each processor has a memory of constant size and can generate elements without the need to deal with large integers such as $B(n)$.

5. Generating binary trees with parent-children relations

If binary trees are generated using the inversion table representation, then the parent-children relations are determined as follows. If $x_{i-1}+1=x_i$ then node *i* is the left child of node $i-1$. Otherwise *i* is the right child of a node *j*, where *j* is the maximal index such that $j<i$ and $x_j=x_i$. For example, consider the sequence 001201 (see Fig. 1). Node 2 (3,4,5,6) is the right (left, left, right, left) child of node 1 (2,3,2,5, respectively). The parallel generating algorithm can be extended to provide the addresses of parent and children nodes for each node, using the EREW PRAM model of computation. The solution is given in the full version of the paper.

By contrast, if the new BFS parent sequence notation is used, then each node knows the address of its parent, directly on a linear array with *n* processors. However, learning the addresses of the children of each node requires EREW PRAM model. The same is valid if actual access to either parent or childen is sought once the tree is generated. But, the corresponding extension of the algorithm is rather trivial, and is already included as part of sequential algorithm in section 3. We repeat this trivial instruction here because of a minor modification being made.

```
for $i \leftarrow 2$ to $n$ do {$k \leftarrow p_i/2$; $parent_i \leftarrow k$; if $p_i$ even then $leftchild_k \leftarrow i$ else $rightchild_k \leftarrow i$}
```

This instruction is much simpler than the method used for the inversion table representation, the latter requiring additional variables, some background explanation, and a tricky implementation. Therefore, we obtain the following result.

Theorem 2. The parallel algorithm described above generates all trees with *n* nodes (each node being associated with its parent and children) in lexicographic order as defined by BFS parent array notation, with constant delay per tree. The algorithm achieves an optimal cost of $O(n*B(n))$, and trees are generated without the need to deal with large integers like $B(n)$. If the address of parent of each node is sufficient then the algorithm runs on a linear array of *n* processors and constant size memory per processor, while the address and access to each parent and child are provided on EREW PRAM with *n* processors and shared memory of size $O(n)$.

6. A notation for fast access to children on a linear array

As pointed out in section 5, the BFS parent sequence notation allows each node to determine the address of its parent in constant time on a linear array of *n* processors. By contrast, a significantly more powerful model of computation, namely the EREW PRAM, appears to be needed in order to determine the children of a node in constant time, using the same notation. In this section we introduce a new representation of binary trees which allows constant time determination of a node's children on a linear array of $n-1$ processors. We then show how this new notation can be used to generate all binary trees with n nodes by an algorithm satisfying all desirable properties listed in section 1.

As with the notation of section 3, we use a left-to-right BFS labeling of trees. All nodes are labeled by consecutive integers $1, 2, \ldots, n$ such that nodes on a lower level are labeled before those on a higher level, while nodes on the same level are labeled from left to right, as in Fig. 3. A tree is represented using a children array $c_2, c_3, \ldots, c_{2n-1}$ as follows: the left and right children (if any) of node i are stored in c_{2i} and c_{2i+1}, respectively; missing children are denoted by 0. As an example, the children array representation of the tree in Fig. 3 is 0234506000; the next four trees to be generated are 0234500600, 0234500060, 0234500006, and 0234056000. Note that the root has no parent, and node n has no children.

Let $c_1=1$ and $c_{2n}=0$. Clearly the array c contains elements $1, 2, \ldots, n$ with n zeros inserted in between. Because of the left-to-right BFS numbering, if node i is the parent of node j then $i<j$, and therefore either $c_{2i}>i$ or $c_{2i+1}>i$; in other words, $c_i>i/2$ (if positive). Define $b_i=1$ if $c_i>0$ and $b_i=0$ otherwise. Then $b_1+\ldots+b_i = max(c_1, \ldots, c_i) \geq i/2$. Thus the array b satisfies the prefix property. This means that there is an easy relation between bitstring sequences that satisfy the prefix property and the children arrays. However, the corresponding trees in the bitstring representation [33,51] and children array representation are different. Nevertheless the same algorithms to generate them as tree sequences may be used (with minor modification). For bitstring sequences sequential algorithms are given in [10,11]. Here we give a simpler algorithm to generate children array tree sequences.

The rightmost element of array c that can be occupied by an integer $j>0$, representing node j, is c_{2j-1}. We say that an integer j is *mobile* if it is not in c_{2j-1} and all (non-zero) integers to its right occupy their rightmost positions. A simple sequential algorithm that uses this notation to generate all binary trees with n nodes is given below. Initially, $c_i=i$, for all $i \leq n$, and $c_i=0$, for all $i > n$. If numerical order $0 < 1 < \ldots < n$ is assumed, the algorithm generates children array sequences in reverse lexicographic order. Alternatively, the order may be interpreted as lexicographic if 0,1,...,n are treated as symbols, ordered as ‘1’ < ‘2’ <... < ‘*n*’ < ‘0’. Numeric lexicographic order may be obtained if 0 is replaced by a number larger than n (the algorithm should always report that number instead of 0).

```
while there is a mobile integer $c_i$ do { $c_{i+1} \leftarrow c_i$; $c_i \leftarrow 0$;
  for $k \leftarrow 1$ to $n - c_{i+1}$ do $c_{i+k+1} \leftarrow c_{i+k} + 1$; for $k \leftarrow i + n - c_{i+k} + 2$ to $2n - 1$ do $c_k \leftarrow 0$}
```

This representation can be used directly to store a binary tree on a linear array of $n-1$ processors: processor i stores c_{2i} and c_{2i+1}; by inspecting these two values, processor i can readily determine two children of node i. We show in the full version of the paper how to simulate the above sequential algorithm on a linear array of processors.

If binary trees are to be generated on a linear array of processors so that each node's parent can be obtained quickly, then the algorithm of section 4 should be used. On the other hand, if fast access to a node's children is important, then the algorithm of this section becomes more attractive. If both forms of access are needed then both algorithms can be executed simultaneously since they generate the same tree at each step (it can be easily verified that the parent and children arrays generate trees in the same order). This proves the following theorem.

Theorem 3. By combining the parent and children array representations of binary trees, it is possible to generate binary trees (with parent-children relations) in parallel such that properties (1)-(6) are preserved.

A sequence of n left parentheses and n right parentheses is *well formed* if the number of right parentheses encountered when scanning from left to right never exceeds the number of left parentheses. It is known that these sequences are in one-to-one correspondence with binary trees on n nodes [21]. Given a binary tree in the above notation, the following transformation can be used to obtain the corresponding well formed parenthesis sequence: Replace every 0 element with a '(' and every element >0 with a ')'. Thus, for example, the 6-node binary tree given by $c_1 c_2 \dots c_{12}$ = 102345060000, yields the parenthesis sequence ()(((()()))). It follows therefore that the above mentioned sequential and parallel (detailed in the full version of the paper) algorithms can be used to generate well formed sequences of parentheses.

7. Generating *t*-ary trees

In this section we generalize our BFS notations, sequential and parallel algorithms to the case of t-ary trees. A tree is a t-ary tree iff each of its nodes has at most t children. There is one-to-one correspondence between t-ary trees and regular t-ary trees in which each internal node has exactly t children; the correspondence is obtained in a similar way as for binary trees. Therefore, all results here apply for regular t-ary trees.

The number of t-ary trees with n nodes is $(tn)!/(n!(tn-n)!)/((t-1)n+1)$ (cf.[19,51]). Sequential algorithms for generating t-ary sequences appear in [7,11,12,36,37,42,48,50,51].

We apply the same BFS labeling for t-ary trees (see Fig. 4 for an example of a 3-ary tree; dashed edges denote non-existing children). The parrent array $p_1 \dots p_n$ is defined as follows:

$p_1=1, p_i=t(j-1)+l+1$ if i is the l-th child of node j, $2 \le i \le n$.

For example, the 3-ary tree on Fig. 4 has parent array 1,2,3,4,8,10,18. Obviously, it follows that $p_{i-1} < p_i \le ti-t+1$ for $2 \le i \le n$. Therefore the sequences $p_1 \dots p_n$ are defined in the same way as z-sequences from [51]. However, the same sequence refers to different t-ary trees in these two notations. Nevertheless, the sequential algorithm [51] for generating z-sequences can be used to generate BSF parent sequences in our notation. The algorithm is extended from [51] to include parent-children relations, and is a generalization of the algorithm in section 3. The l-th children of node i is denoted by $child_{i,l}$ (it is 0 if no such child exist) while $parent_i$ denote the parent node of i. Integer division is used throughout the algorithm.

```
for $i \leftarrow 1$ to $n$ do  for $l \leftarrow 1$ to $t$ do $child_{i,l} \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do {$p_i \leftarrow i; parent_i \leftarrow (i-2)/t+1; l \leftarrow p_i-1-t(parent_i-1)$; $child_{(i-2)/t+1,l} \leftarrow i$ }
repeat
    report $t$-ary tree; $j \leftarrow n$;
    while $p_j<2j-1$ and $j>1$ do {$i \leftarrow parent_j; l \leftarrow p_i-1-t(i-1)$;  $child_{i,l} \leftarrow 0; j \leftarrow j-1$};
    $p_j \leftarrow p_j+1$; for $i \leftarrow j+1$ to n do $p_i \leftarrow p_{i-1}+1$;
    for $i \leftarrow j$ to $n$ do {$k \leftarrow (p_i-2)/t+1; parent_i \leftarrow k$;  $l \leftarrow p_i-1-t(k-1)$; $child_{k,l} \leftarrow i$ }
until $p_1=2$
```

Let $x_i+p_i = ti-t+1$. Then $x_1=0$ and $0 \le x_i \le x_{i-1}+t-1$ for $2 \le i \le n$. Sequence $x_1 \dots x_n$ is the generalization of the analogous one which we used for binary trees, and will serve for designing a parallel algorithm for generating t-ary tree sequences on a linear array of processors. The t-ary tree sequences generation algorithm can be obtained from the corresponding one for binary trees in section 4 by the following two modifications:

-in the first if statement replace the condition $x_i \le x_{i-1}$ with the condition $x_i \le x_{i-1}+t-2$,

-in the sixth if statement replace the assignment $w_n \leftarrow x_{n-1}+2$ with $w_n \leftarrow x_{n-1}+t$.

The inclusion of parent children relations can be done similarly as in the case of binary trees, using the following instruction.

```
for $i \leftarrow 2$ to $n$ do {$k \leftarrow (p_i-2)/t+1$; $parent_i \leftarrow k$;  $l \leftarrow p_i-1-t(k-1)$; $child_{k,l} \leftarrow i$ }
```

The children array notation presented in section 6 also extends easily to represent t-ary trees. The prefix property of bitstring sequences $b_1 \ldots b_{tn}$ for t-ary trees is as follows: the number of 0's is always less than or equal to $t-1$ times the number of 1's while scanning from b_1 to b_{tn}. The parallel generating algorithm can be generalized using the following result [11]: The number of bitstring sequences $b_{i+1} \ldots b_{tn}$ containing q ones and $tn-q-i$ zeros such that there exist a sequence $b_1 \ldots b_{tn}$ satisfying the prefix property is $(t(n-q)-i+1)/(tn-i-q+1)\ C(tn-i,q)$.

The parallel algorithm for t-ary trees is obtained by introducing minor modifications to the one for binary trees. In particular, the size of the corresponding counters should be provided. These counters have similar properties with respect to variable q, as in the case of binary trees.

8. Conclusion

The parallel algorithms presented in this paper can be made adaptive (i.e. to run on a parallel model of computation consisting of an arbitrary number k of processors) if the processors are divided into k/n groups of n processors each such that each group produces an interval of consecutive trees. The first and the last tree in each group can be determined in a preprocessing step by applying any known unranking function [51,48,41,10,11] that follows the lexicographic order of z-sequences of trees. However, these function involve very large integers. Another scheme that does not deal with large integers and yet divides the job evenly among the groups is described in [47]. Sequential algorithms exist for generating AVL trees [23], 2-3 trees [15], B-trees [16], non-regular trees [13], unordered trees [30], free trees [49], trees with n nodes and m leaves [31], trees with nodes of any degree [44], and trees with bounded height [22]. It remains an open problem to generate parallel algorithms for generating these kinds of trees.

References

[1] S.G. Akl, The Design and Analysis of Parallel Algorithms, Prentice Hall, 1989.
[2] S.G. Akl, Adaptive and optimal parallel algorithms for enumerating permutations and combinations, *Comp. J.*, 30, 5, 433-436, 1987.
[3] S.G. Akl, D. Gries, and I. Stojmenovic, An optimal parallel algorithm for generating combinations, *Information Processing Letters,* Vol. 33, 1989/90, 135-139.
[4] S.G. Akl, J. Calvert, and I. Stojmenovic, Systolic generation of derangements, Algorithms and Parallel VLSI Architectures II, Elsevier, 1992, 59-70.
[5] S.G. Akl, H. Meijer, and I. Stojmenovic, An optimal parallel algorithm for generating permutations in lexicographic order, *J. of Parallel and Distributed Computing,* to appear.
[6] S.W. Bent, Ranking trees generated by rotations, Proc. SWAT 1990, 132-142.
[7] T. Beyer, and S. M. Hedetniemi, Constant time generation of rooted trees, *SIAM J. Comput.*, 9, 4, 1980, 706-712.
[8] B. Chan and S.G. Akl, Generating combinations in parallel, *BIT,* 26, 1, 2-6, 1986.
[9] G.H. Chen and M.-S. Chern, Parallel generation of permutations and combinations, *BIT* 26, 277-283, 1986.
[10] M.C. Er, Enumerating ordered trees lexicographically, *Comp. J.*, 28,5, 538-542, 1985.
[11] M.C. Er, Lexicographic listing and ranking t-ary trees, *Comp. J.*, 30, 6, 1987, 569-572.
[12] M.C. Er, Lexicographic enumeration of k-ary trees, *J. Inf. Opt. Sci.*, 6, 1985, 137-145.
[13] M.C. Er, A simple algorithm for generating non-regular trees in lexicographic order, *Comp. J.*, 31, 1988, 61-64.
[14] D.K. Gupta, On the generation of p-sequences, *Intern. J. Comp. Math.*, 38, 31-35, 1991.
[15] U.I. Gupta, D.T. Lee, and C.K. Wong, Ranking and unranking of 2-3 trees, *SIAM J. Comput.*, 11, 1982, 582-590.
[16] U.I. Gupta, D.T. Lee, and C.K. Wong, Ranking and unranking of B-trees, *J. of Alg.*, 4,

1983, 51-60.
[17] T. Hikita, Listing and counting subtrees of equal size of a binary tree, *Inf. Proc. Lett.*, 17, 1983, 225-229.
[18] G.D. Knott, A numbering system for binary trees, *Comm. of ACM*, 20, 1977, 113-115.
[19] D.E. Knuth, The Art of Computer Programming, Vol. 1, Addison-Wesley, 1968.
[20] D.E. Knuth, Lexicographic permutations with restrictions, *Discr. Appl. Math.*, 1, 1979, 117-125.
[21] R.L. Kruse, Data Structures and Program Design, Prentice Hall, New Jersey, 1984.
[22] C.C. Lee, D.T. Lee, and C.K. Wong, Generating binary trees of bounded height, *Acta Inf.*, 23, 1986, 529-544.
[23] L. Li, Ranking and unranking of AVL trees, *SIAM J. Comp.*, 15, 1986, 1025-1035.
[24] C.J. Lin, Parallel generation of permutations on systolic arrays, *Par. Comp.*, 15, 1, 1990, 267-276.
[25] E. Makinen, A survey on binary tree codings, *Comput. J.*, 34, 5, 1991, 438-443.
[26] E. Makinen, Efficient generation of rotational-admissible codewords for binary trees, *Comput. J.*, 34, 4, 1991, 379.
[27] H.W. Martin, J.O. Kiltinen, and R.B. McNeill, The representation and enumeration of ordered k-ary trees, Proc. of *Int. Conf. on Computing and Information,* 1989, 105-109.
[28] H.W. Martin and B.J. Orr, A random binary tree generator, Computing Trends in the 1990's, *ACM 17th Computer Science Conference*, Louisville, Kentucky, Feb. 1989, 33-38.
[29] J.M. Pallo, Enumerating, ranking, unranking binary trees, *Comp. J.*, 29, 171-175, 1986.
[30] J.M. Pallo, Lexicographic generation of binary unordered trees, *Patt. Rec. Lett.*, 10, 1989, 217-221.
[31] J.M. Pallo, Generating trees with n nodes and m leaves, *Int. J. Comp. Math.*, 21, 1987, 133-144.
[32] J.M. Pallo, R. Racca, A note on generating binary tree in A-order and B-order, *Intern. J. Comp. Math.*, 18, 1985, 27-39.
[33] A. Proskurowski, On the generation of binary trees, *J. ACM,* 27, 1-2, 1980.
[34] A. Proskurowski, F. Ruskey, Binary tree gray codes, *J. Alg.*, 6, 225-238, 1985.
[35] A. Proskurowski, F. Ruskey, Generating binary trees by transpositions, SWAT 1988, *Lect. Notes Comp. Sci.*, 318,199-207.
[36] D. Roelants van Baronaigien and F. Ruskey, Generating t-ary trees in A-order, *Inf. Process. Lett.*, 27, 1988, 205-213.
[37] D. Roelants van Baronaigien and F. Ruskey, A Hamiltonian path in the rotation lattice of binary trees, *Congressus Numerantium* 59, 1987, 313-318.
[38] D. Rotem and Y.L. Varol, Generation of binary trees from ballot sequences, *J. ACM*, 25, 396-404, 1978.
[39] F. Ruskey, Listing and counting subtrees of tree, *SIAM J. Comp.*, 10, 1, 1981, 141-150.
[40] F. Ruskey, Generating t-ary trees lexicographically, *SIAM J. Comp.*, 7, 1978, 492-509.
[41] F. Ruskey and T.C. Hu, Generating binary trees lexicographically, *SIAM J. Comp.*, 6, 1977, 745-758.
[42] H. Scoins, Placing trees in lexicographic order, *Machine Intelligence,* 3, 1969, 43-60.
[43] I. Semba, Generation of binary trees from stack permutations, *J. of Inf. Proc.*, 5, 2, 102-105, 1982.
[44] W. Skarbek, Generating ordered trees, *Theor. Comp. Sci.* 57, 1988, 153-159.
[45] M. Solomon and R.A. Finkel, A note on enumerating binary trees, *J. ACM* 27, 1980, 3-5.
[46] I. Stojmenovic, An optimal algorithm for generating equivalence relations on a linear array of processors, *BIT*, Vol. 30, 3, 1990, 424-436.
[47] I. Stojmenovic, On random and adaptive parallel generation of combinatorial objects, *Int. J. Computer Math.*, 42, 1992, 125-135.
[48] A.E. Trojanowski, Ranking and listing algorithms for k-ary trees, *SIAM J. Comp.*, 7, 4, 1978, 492-509.
[49] R.A. Wright, B. Richmond, A. Odlyzko, and B.D. McKay, Constant time generation of free trees, *SIAM J. Comp.*, 15, 1986, 540-548.
[50] S. Zaks and D. Richards, Generating trees and other combinatorial objects lexicographically, *SIAM J. on Comput.*, 8, 1, 1979, 73-81.
[51] S. Zaks, Lexicographic generation of ordered trees, *Theor. Comp. Sci.*, 10, 63-82, 1980.
[52] S. Zaks, Generation and ranking of k-ary trees, *Inf. Proc. Lett.*, 14, 1, 1982, 44-48.
[53] D. Zerling, Generating binary trees using rotations, *J. ACM,* 32, 694-701, 1985.

An efficient parallel partitioning algorithm

R.Sarnath
Deptt of Computer Science
226, Bell Hall
SUNY at Buffalo
Amherst NY 14260
email: sarnath@cs.buffalo.edu
Phone (716) 645-3774

Abstract

The problem of finding the coarsest partition of a set S with respect to another partition of S and a function on S has several applications. In this paper, present efficient parallel algorithms for the problem.

Keywords: Parallel-algorithms, Partitioning

1 Introduction

The **single function coarsest partitioning** problem can be stated as follows:

Problem : *Given a set S of n elements, $a_1, a_2, ..., a_n$, a partition $\pi = (\pi_1, \pi_2, ..., \pi_k)$ of S and a function $f : S \rightarrow S$, we wish to find the* **coarsest** *(i.e., having the fewest blocks) partition $\pi' = (\pi'_1, \pi'_2, ..., \pi'_l)$ of S, such that*

1. $\forall a_i, a_j \in S, a_i, a_j \in \pi'_m \Rightarrow \exists p$ *such that,* $f(a_i), f(a_j) \in \pi'_p$.

2. $\forall m, 1 \leq m \leq l, \exists p, 1 \leq p \leq k,$ *such that* $\pi'_m \subseteq \pi_p$.

If $a_i \in \pi_m$, we then say that m is the **initial label** of a_i. If $a_i \in \pi'_p$, we then say that p is the **final label** of a_i. We shall refer to a_i as the i-th element, or the element with index i. The more general version of the problem, is the **multiple function coarsest partition** problem, where we are given c, $c \geq 1$, functions $f_1, f_2, ..., f_c$. All the c functions must satisfy condition 1 above. This problem has several interesting applications, some of the important ones being *minimizing the number of states in a DFA*, *checking isomorphism of DFAs* and *deciding isomorphism of triply connected planar graphs*. Given a DFA, we start with the initial partition comprising of

2 blocks - the set of final, and non-final states. The functions are the state transitions on each symbol of the input alphabet. The final partition yields the minimum DFA.

The first $O(n \log n)$ algorithm for the problem was given in [1]. This algorithm successively refines the partition, by examining the inverse image of the elements under the given mappings. It does not appear that this approach can be easily parallelised. An alternate $O(n \log n)$ algorithm for the problem was given by Rao et al [6], using an inverted list implementation. Paige et al [5] give an $O(n)$ time algorithm for the single function case. Their approach exploits the structure of the mapping, and does not use the idea of successive refinements. Parallel algorithms for these problems have been given by Ja Ja and Kosaraju [3]. Their approach yields an n^2 processor, $O(log^2 n)$ time algorithm for the single function case, and an n^4 processor, $O(log^2 n)$ time algorithm for the multiple function problem. These algorithms run on the CREW PRAM [4]. Their algorithm for the single function case can be implemented on a $\sqrt{n} \times \sqrt{n}$ mesh in $O(\sqrt{n})$ time. In this paper we use a simple yet interesting property of the partitions (stated in lemma 1), to derive algorithms for the single function partitioning problem, that run in time $O(\log n)$ using n^2 processors on a CREW PRAM, and in $O(log^2 n)$ time using n processors on an EREW PRAM [4]. Our algorithm is thus within a factor of $log^2 n$ of the optimal.

2 The algorithm

The following lemma tells us something interesting about the final partition.

Lemma 1: *Given any two elements $a_i, a_j \in S$, a_i and a_j will belong to different blocks in the final partition, if and only if, $\exists k, 0 \leq k \leq n(n-1)/2$, such that, $f^k(a_i)$ and $f^k(a_j)$ belong to different initial blocks. (where $f^0(a) = a$, and $f^k(a) = f(f^{k-1}(a))$.)*

Proof : We shall first prove that if any k exists such that $f^k(a_i)$ and $f^k(a_j)$ belong to different initial blocks, then there must be a k smaller than $n(n-1)/2$ such that $f^k(a_i)$ and $f^k(a_j)$ belong to different initial blocks.

To see this, consider all unordered pairs of elements of S. There are only $n(n-1)/2$ of these. If we examine all pairs $(f^k(a_i), f^k(a_j))$ for k in the above range, we have definitely repeated a pair (by the pigeonhole principle). Therefore we do not have to look any further.

(if part) Suppose there exists k satisfying the conditions in the lemma, i.e., $f^k(a_i)$ and $f^k(a_j)$ belong to different initial blocks. Clearly, they must also belong to different final blocks. We can then reason backwards as follows - $f^{k-1}(a_i)$ and $f^{k-1}(a_j)$ belong to different final blocks

as well, and the result follows.

(only if part) Suppose that a_i and a_j belong to different final blocks, π'_l and π'_m respectively, and by way of contradiction, no k satisfying the conditions in the lemma exists. Now, for all elements x, such that $x \in \pi'_l$, and for all values of k, $f^k(x)$ and $f^k(a_i)$ belong to the same initial block. Also, for all elements y, $y \in \pi'_m$, $f^k(y)$ and $f^k(a_j)$ belong to the same initial block, for all values of k. But this would imply that the blocks π'_l and π'_m can be merged into one, and therefore the partitioning π' of S is not the coarsest - a contradiction. □

Our algorithm uses the above fact to successively refine the partition. We first present an algorithm that runs in $O(\log n)$ time using n^2 processors. We use an array A of size n^2, which always reflects the current refinement. At any stage, $A[i, j] = 1$ if and only if a_i and a_j belong to the same block. We use another array P of size n; after step t, $P[i]$ holds the index of $f^{2^t}(a_i)$. The array P is initialised by assigning $P[i]$ the value j, if $f(a_i) = a_j$. The algorithm is given below:

```
Algorithm Partition1
(⋆ Initialize $A$ according to the given initial partition ⋆)
Repeat $2 \log n$ times
For all pairs $(i, j)$, do the following in parallel:
    • If $(A[P[i], P[j]] = 0)$ then $A[i, j] := 0$
    • $P[i] := P[P[i]]$
end Repeat
end Partition1
```

The algorithm clearly runs in time $O(\log n)$ using n^2 processors on a CREW PRAM. We shall now refine this idea to present the second algorithm. Here, we shall store the information about the current refinement by assigning labels to the elements, and updating these labels at every step. We replace the array A of the previous algorithm by an array L of size n, that holds the labels of the elements in S. To begin with, L holds the initial labels; as the algorithm proceeds, the labels are modified at each step. At the end, the final labelling can be found in L. The array P plays the same role as in the earlier algorithm.

```
Algorithm Partition2
(* The arrays are initialised as described above *)
Repeat $2 \log n$ times:
    1. Do the following, for all $i$, in parallel :
        1.1 $L[i] := n * (L[i] - 1) + L[P[i]]$;
        1.2 $P[i] := P[P[i]]$;
    2. Sort the array $L$ and assign $L[i] :=$ Rank of $L[i]$ in $L$.
end Repeat
end Partition2
```

Note that while reassigining labels in the first step of the repeat loop, we must be careful that elements of different partitions are not assigned the same label. The formula we have takes care of this. The correctness of the algorithm follows from the following lemma.

Lemma 2: *For all pairs* (a_i, a_j), a_i *and* a_j *have the same label after* l *executions of the repeat loop, if and only if,* $\forall m, 0 \leq m \leq 2^l - 1$, $f^m(a_i)$ *and* $f^m(a_j)$ *belong to the same initial partition.*

Proof: The argument proceeds by induction on l. The base case ($l = 0$) is obvious. By induction hypothesis, after $l - 1$ steps, a_i and a_j have the same label, if and only if, $\forall m$, $0 \leq m \leq 2^{l-1} - 1$, $f^m(a_i)$ and $f^m(a_j)$ belong to the same initial block. Let $f^{2^{l-1}}(a_i) = a_p$ and $f^{2^{l-1}}(a_j) = a_q$. Once again, by induction hypothesis, after $l - 1$ steps, a_p and a_q have the same label, if and only if, $\forall m$, $0 \leq m \leq 2^{l-1} - 1$, $f^m(a_p)$ and $f^m(a_q)$ belong to the same initial block. From the structure of the algorithm, it is clear that a_i and a_j will have the same label after the l-th step, if and only if,

- a_i and a_j have the same label at the end of step $l - 1$, *and*
- a_p and a_q have the same label at the end of step $l - 1$.

Now the result of the lemma follows from the observation that

$f^{2^{l-1}+m}(a_i) = f^m(a_p)$ and $f^{2^{l-1}+m}(a_j) = f^m(a_q)$. □

We therefore have the following theorem:

Theorem 1: *The algorithm Partition2 computes the coarsest possible partition subject to the given constraints in time* $O(\log^2 n)$ *using* n *processors on an EREW PRAM.*

Proof: From lemma 2, we have that after $2 \log n$ steps of the repeat loop, a_i and a_j will have the same label, if and only if,
$\forall m, 0 \leq m \leq (n^2 - 1)$, $f^m(a_i)$ and $f^m(a_j)$ have the same initial label (i.e., belong to the same initial block). Lemma 1 then assures us that we indeed have the coarsest possible partition.

Clearly all the assignment statements in the first step of the repeat loop can be done by n processors on a CREW PRAM in $O(1)$ time, and therefore in $O(\log n)$ time on an EREW PRAM [8]. The sorting can be done in $O(\log n)$ time using n processors [2]. Since the loop executes $O(\log n)$ times, we have the required result. □

3 Conclusion

The best known sequential algorithm for the problem has a worst case running time $O(n)$. Our parallel algorithm is thus within a $\log^2 n$ factor of the optimal. Even if we used a stronger PRAM model, the sorting we do at each step appears unavoidable. (Otherwise, the sizes of the labels would be $O(n^{\log n})$, i.e. $O(\log^2 n)$ bits long.) This looks like a fundamental bottleneck that prevents us from getting an optimal parallel algorithm. It would be an interesting open question to see if this bottleneck can be broken.

References

[1] A.V.Aho, J.E.Hopcroft and J.D.Ullman, The design and analysis of computer algorithms, Addison-Wesley, 1974.

[2] R.Cole, Parallel merge sort. Proc. 27th annual IEEE FOCS, pp 511-516, 1986.

[3] J.Ja' Ja' and S.R.Kosaraju, Parallel algorithms for planar graph isomorphism and related problems, IEEE trans. on circuits and systems, vol 35, 3, March 1988, pp 304-310.

[4] R.Karp and V.Ramachandran, A survey of parallel algorithms for shared memory architectures. Report No. UCB/CSD 88/408, March 1988.

[5] R.Paige, R.E.Tarjan and R.Bonic, A linear time solution to the single function coarsest partition problem, Theoretical computer science, 40, 1985, pp67-84.

[6] N.Rao, R.Sridhar and S.S.Iyengar, An $O(n \log n)$ set refinement algorithm with applications, Intl journal of computer math, vol 40, 1991, pp 129-138.

[7] C.D.Thompson and H.T.Kung, Sorting on a mesh connected computer, Comm of the ACM, 20(4), 1977, pp 263-271.

[8] U.Vishkin, Implementation of simultaneous memory address access in models that forbid it. J. Algorithms, vol 4, 1983, pp45-50.

NUMERICAL SOLUTION OF GENERALIZED LYAPUNOV EQUATION FOR DESCRIPTOR SYSTEM

Pradeep Misra
Department of Electrical Engineering
Wright State University
Dayton, OH 45435
`pmisra@valhalla.cs.wright.edu`

Abstract: Numerical solution of continuous as well as discrete time Lyapunov matrix equation for state space systems is very well studied. Excellent numerical technique for their solutions also exist. However, these results do not seem to have been extended to the Lyapunov matrix equations arising in the study of generalized state space systems. This paper attempts to address this situation and proposes a possible numerical approach for the numerical solution of generalization of Lyapunov equations for descriptor systems.

1. Introduction

It is well known that for linear multivariable systems described by state space equations

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) \end{aligned} \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$, the matrix Lyapunov equation that arises is the study of stability of (1.1) or in computation of controllability and observability Gramians [1]-[5], is given by:

$$\begin{aligned} A^T P + PA &= -Q \quad \text{or} \\ AP + PA^T &= -Q \end{aligned} \tag{1.2}$$

Note that Q is any symmetric positive semi-definite matrix to study the stability of (1.1) and assumes the values BB^T and $C^T C$ respectively for computation of controllability and observability Gramians of the system. An analogous matrix equation for discrete time systems

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) + D\mathbf{u}(k) \end{aligned} \tag{1.3}$$

is given by

$$\begin{aligned} A^T PA - P &= -Q \quad \text{or} \\ APA^T - P &= -Q \end{aligned} \tag{1.4}$$

where Q is again a symmetric positive semi-definite matrix. The solution of the matrix Lyapunov equations (1.2) and (1.4) are unique, positive definite matrices, provided that the state matrices in the two cases are stable.

Numerically, the problem of computation of P matrices for both continuous and discrete time Lyapunov matrix equations are solved by transforming the state matrices to their respective real Schur form and recursively determining the elements of P. Reliable numerical algorithms exist for their solution [6]-[8].

In this paper we extend the above results to the class of systems described by their generalized state space equations:

$$\begin{aligned} E\dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) \end{aligned} \tag{1.5}$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{u} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^p$ for continuous time systems and by

$$\begin{aligned} E\mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) \\ \mathbf{y}(k) &= C\mathbf{x}(k) + D\mathbf{u}(k) \end{aligned} \tag{1.6}$$

for discrete time systems. The dimensions of state, input and output variables is assumed to be the same for discrete discrete time systems.

Note that in (1.5) and (1.6), if the descriptor matrix E has full rank then pre-multiplying the state equation by E^{-1}, one can reduce (1.5) and (1.6) to state equations similar to (1.1) and (1.3) and solve the corresponding (modified) Lyapunov equations (1.2) and (1.4). However, if E is close to being a singular matrix or numerically ill-conditioned with respect to inversion, then the solution of the modified Lyapunov equations may not be reliable. In this paper, it will be assumed that while E is non-singular, it is poorly conditioned with respect to inversion.

2. Lyapunov Equations for Generalized Systems

In this section the Lyapunov equations for continuous as well as discrete systems will be developed from the first principles. For the continuous time dynamical equations $E\dot{\mathbf{x}}(t) = A\mathbf{x}(t)$ as well as the discrete time equation $E\mathbf{x}(k+1) = A\mathbf{x}(k)$, assume that the Lyapunov function is defined as

$$\begin{aligned} v(t) &= \mathbf{x}^T(t)E^TPE\mathbf{x}(t) \quad \text{or} \\ v(k) &= \mathbf{x}^T(k)E^TPE\mathbf{x}(k) \end{aligned} \tag{2.1}$$

where $P > O$ and E has full rank. Under these assumptions, clearly $v(t)$ is a positive scalar function.

2.1. Continuous Time Systems

With the Lyapunov function defined as in (2.1), consider

$$\frac{dv(t)}{dt} = \frac{d\mathbf{x}^T(t)}{dt}E^TPE\mathbf{x}(t) + \mathbf{x}^T(t)E^TPE\frac{d\mathbf{x}(t)}{dt}. \tag{2.2}$$

Now,

$$\frac{d\mathbf{x}^T(t)}{dt}E^T = \left(E\frac{d\mathbf{x}(t)}{dt}\right)^T = (A\mathbf{x}(t))^T = \mathbf{x}^T(t)A^T \tag{2.3}$$

On substituting (2.3) in (2.2), we get

$$\begin{aligned} \frac{dv(t)}{dt} &= \mathbf{x}^T(t)A^TPE\mathbf{x}(t) + \mathbf{x}^T(t)E^TPA\mathbf{x}(t) \\ &= \mathbf{x}^T(t)[A^TPE + E^TPA]\mathbf{x}(t) \\ &= -\,\mathbf{x}^T(t)[EQE^T]\mathbf{x}(t) \quad \text{say} \end{aligned} \tag{2.4}$$

where, $Q \geq O$. Note that the right hand side of the last equality in (2.4) has been selected to make these matrix equations compatible with the case of standard state space systems.

Therefore, the matrix Lyapunov equation for continuous time systems is

$$A^T PE + E^T PA + EQE^T = O \tag{2.5}$$

Note that when $E = I$ (standard state space systems), (2.5) reverts to the corresponding Lyapunov equation (1.2).

2.2. Discrete Time Systems

With $v(t) = \mathbf{x}^T(k)E^T PE\mathbf{x}(k)$, consider the increment in Lyapunov function:

$$\begin{aligned}\Delta v(\mathbf{x}(k)) &= v(x(k+1)) - v(x(k)) \\ &= \mathbf{x}^T(k+1)E^T PE\mathbf{x}(k+1) - \mathbf{x}^T(k)E^T PE\mathbf{x}(k)\end{aligned} \tag{2.6}$$

However,

$$\mathbf{x}^T(k+1)E^T = (E\mathbf{x}(k+1))^T = (A\mathbf{x}(k))^T = \mathbf{x}^T(k)A^T \tag{2.7}$$

Substituting (2.7) in (2.6), we get

$$\begin{aligned}\Delta v(\mathbf{x}(k)) =& \mathbf{x}^T(k)A^T PA\mathbf{x}(k) - \mathbf{x}^T(k)E^T PE\mathbf{x}(k) \\ =& \mathbf{x}^T(k)[A^T PA - E^T PE]\mathbf{x}(k) \\ =& - \mathbf{x}^T(k)[EQE^T]\mathbf{x}(k)\end{aligned} \tag{2.8}$$

where, again $Q \geq O$ and the right hand side of the last equality in (2.8) has been selected to make these matrix equations compatible with the case of standard discrete time state space systems.

The matrix Lyapunov equation for discrete time systems from (2.8) is

$$A^T PA - E^T PE + EQE^T = O \tag{2.9}$$

Further, if $E = I$, one obtains the Lyapunov equation (1.4) for standard discrete time state space systems.

3. Numerical Solution of Generalized Lyapunov Equations

Theoretically, it is possible to solve the above generalized Lyapunov equations by transforming the problem of determination of P to solving linear algebraic equations as follows: Using Kronecker products, it can be shown that

$$\operatorname{vec}(XYZ) = (Z^T \otimes X)\operatorname{vec}(Y) \tag{3.1}$$

where,

$$\operatorname{vec}(Z) = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_n \end{bmatrix} \tag{3.2}$$

and $\mathbf{z}_i$ is the i-th the columns of the matrix Z [9]. Applying the transformation to (2.5) and (2.9), the equivalent representation of the continuous time and discrete time generalized Lyapunov equations become

$$\begin{aligned}(E^T \otimes A^T + A^T \otimes E^T)\,\mathrm{vec}(P) &= -\,\mathrm{vec}(\hat{Q}) \quad \text{and} \\ (A^T \otimes A^T - E^T \otimes E^T)\,\mathrm{vec}(P) &= -\,\mathrm{vec}(\hat{Q})\end{aligned} \tag{3.3}$$

respectively where $\hat{Q} = EQE^T$. Therefore, in order to solve for n-th order generalized Lyapunov equation, one will need to solve a system of linear algebraic equations of order n^2. Apart from high computational cost, this approach may also suffer from severe numerical rounding errors as a result of forming the Kronecker products.

Instead of following the above approach, we suggest the use of generalized eigenvalue decomposition [10], [11]. It is well known that given a square matrix pencil $(A - \lambda E)$, there always exist orthogonal matrices U and V of compatible dimension such that

$$\Delta_E = U^T E V, \quad \Delta_A = U^T A V \tag{3.4}$$

where Δ_E is an upper triangular matrix and Δ_A is a quasi-upper triangular matrix with blocks of dimension at most (2×2) corresponding to complex-conjugate generalized eigenvalues appearing along the diagonal. For the sake of simplicity of presentation, we will assume that Δ_A is strictly upper triangular. The general case will be presented is a more complete version of this paper.

3.1. Continuous Time Matrix Equations

From (3.4),

$$E = U\Delta_E V^T, \quad A = U\Delta_A V^T \tag{3.5}$$

Substituting the expressions for E and A defined by (3.5) into (2.5), we get

$$V\Delta_A^T U^T P U \Delta_E V^T + V\Delta_E^T U^T P U \Delta_A V^T = -\tilde{Q} \tag{3.6}$$

where $\tilde{Q}$ is EQE^T. Next, pre- and post-multiplying (3.6) by V^T and V respectively, we get

$$\Delta_A^T \hat{P} \Delta_E + \Delta_E^T \hat{P} \Delta_A = -\hat{Q} \tag{3.7}$$

where $\hat{P} = U^T P U$ and $\hat{Q} = V^T \tilde{Q} V$. Since Δ_E and Δ_A are both upper triangular, (3.7) can be solved for $\hat{P}$ and therefore for P in a recursive manner. For the purpose of illustration, consider the following (2×2) case: Let,

$$\Delta_A = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix}, \quad \Delta_E = \begin{bmatrix} e_{11} & e_{12} \\ 0 & e_{22} \end{bmatrix}$$

$$\hat{P} = \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \quad \text{and} \quad \hat{Q} = \begin{bmatrix} q_1 & q_2 \\ q_2 & q_3 \end{bmatrix}$$

Then,

$$\Delta_A^T \hat{P} \Delta_E + \Delta_E^T \hat{P} \Delta_A = -\hat{Q}$$

$$\begin{bmatrix} a_{11} & \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} \\ & e_{22} \end{bmatrix} + \begin{bmatrix} e_{11} & \\ e_{12} & e_{22} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ & a_{22} \end{bmatrix} = -\begin{bmatrix} q_1 & q_2 \\ q_2 & q_3 \end{bmatrix} \tag{3.8}$$

On carrying out the above matrix product, we get

$$\begin{bmatrix} 2a_{11}p_1e_{11} & p_1(a_{12}e_{11} + e_{12}a_{11}) + p_2(a_{22}e_{11} + e_{22}a_{11}) \\ \times & 2[p_1a_{12}e_{12} + p_2(a_{22}e_{12} + a_{12}e_{22}) + a_{22}p_3e_{33}] \end{bmatrix} = - \begin{bmatrix} q_1 & q_2 \\ q_2 & q_3 \end{bmatrix} \tag{3.9}$$

where the matrix on the left had side is a symmetric matrix. From (3.9), the elements of $\hat{P}$ can be sequentially computed as

$$\begin{aligned} p_1 &= \frac{-q_1}{2a_{11}e_{11}} \\ p_2 &= \frac{-(q_2 + p_1(a_{12}e_{11} + e_{12}a_{11}))}{a_{11}e_{22} + a_{22}e_{11}} \\ p_3 &= \frac{-(q_3 + 2(a_{12}p_1e_{12} + p_2(a_{22}e_{12} + a_{12}e_{22})))}{2a_{22}e_{22}} \end{aligned} \tag{3.10}$$

The results may be easily generalized for higher order matrix equations.

3.2. Discrete Time Matrix Equations

Once again, substituting the expressions for E and A defined by (3.5) into (2.9), we get

$$V\Delta_A^T U^T P U \Delta_A V^T + V\Delta_E^T U^T P U \Delta_E V^T = -\tilde{Q} \tag{3.11}$$

On pre- and post-multiplying (3.11) by V^T and V respectively, we get

$$\Delta_A^T \hat{P} \Delta_A - \Delta_E^T \hat{P} \Delta_A = -\hat{Q} \tag{3.12}$$

with $\hat{P} = U^T P U$ and $\hat{Q} = V^T \tilde{Q} V$ as before. Using the same notation for $\hat{P}$, $\hat{Q}$, Δ_A and Δ_E as for the continuous time case, the (2×2) matrix Lyapunov equation becomes

$$\Delta_A^T \hat{P} \Delta_A - \Delta_E^T \hat{P} \Delta_E = -\hat{Q}$$

$$\begin{bmatrix} a_{11} & \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ & a_{22} \end{bmatrix} - \begin{bmatrix} e_{11} & \\ e_{12} & e_{22} \end{bmatrix} \begin{bmatrix} p_1 & p_2 \\ p_2 & p_3 \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} \\ & e_{22} \end{bmatrix} = - \begin{bmatrix} q_1 & q_2 \\ q_2 & q_3 \end{bmatrix} \tag{3.13}$$

Explicitly (3.13) becomes

$$\begin{bmatrix} (a_{11}^2 - e_{11}^2)p_1 & p_1(a_{12}a_{11} - e_{12}e_{11}) + p_2(a_{22}a_{11} + e_{22}e_{11}) \\ \times & p_1(a_{12}^2 - e_{12}^2 + 2p_2(a_{22}a_{12} - e_{12}e_{22}) + p_3(a_{22}^2 e_{33}^2 \end{bmatrix} = - \begin{bmatrix} q_1 & q_2 \\ q_2 & q_3 \end{bmatrix} \tag{3.14}$$

Finally, the elements of $\hat{P}$ can be sequentially computed as

$$\begin{aligned} p_1 &= \frac{-q_1}{a_{11}^2 - e_{11}^2} \\ p_2 &= \frac{-(q_2 + p_1(a_{12}a_{11} - e_{12}e_{11}))}{a_{11}a_{22} - e_{22}e_{11}} \\ p_3 &= \frac{-(q_3 + p_1(a_{12}^2 - e_{12}^2) + 2p_2(a_{22}a_{12} - e_{12}e_{22}))}{a_{22}^2 - e_{22}^2} \end{aligned} \tag{3.15}$$

Note that in both continuous as well as discrete case the solution computed was $\hat{P}$. To get the required P, the transformations on $\hat{P}$ must be undone to get $P = U\hat{P}U^T$.

Although the illustrations above are for (2×2) systems, the results can be easily generalized to higher order systems. The elements of P are computed sequencially. The order in which the elements of P are computed is the following: elements $p_{i,1}$, $i = 1, \ldots n$ of the first column are computed first (going down the column). Knowing these elements, the elments of second column $p_{i,2}$, $i = 2, \ldots, n$ can then be computed. In this manner the entire matrix $\hat{P}$ is evaluated.

4. Numerical Examples

Example 1: (Continuous Time Systems) Consider the following system. The matrices represent a continuous time stable system with the descriptor matrix and the state matrices given by

$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \dot{\mathbf{x}}(t) = \begin{bmatrix} -6.1667e^{+00} & -5.0000e^{-01} & -8.3333e^{-01} \\ +5.1667e^{+00} & -2.5000e^{+00} & +1.8333e^{+00} \\ +1.3333e^{+00} & +2.0000e^{+00} & -1.3333e^{+00} \end{bmatrix} \mathbf{x}(t)$$

This example is for the purpose of illustrations only. The matrix E is this case is *not* ill-conditioned with respect to inversion. The generalized eigenvalues of the pencil $(A - \lambda E)$ lie in the left half plane. Selecting

$$EQE^T = \begin{bmatrix} 2.2911e^{+00} & 1.7090e^{+00} & 1.1907e^{+00} \\ 1.7090e^{+00} & 1.3618e^{+00} & 8.9261e^{-01} \\ 1.1907e^{+00} & 8.9261e^{-01} & 9.5121e^{-01} \end{bmatrix},$$

the solution of the continuous time generalized Lyapunov equation was found to be

$$P = \begin{bmatrix} 1.2440e^{-01} & 5.9607e^{-02} & 6.7159e^{-02} \\ 5.9607e^{-02} & 7.6710e^{-02} & 4.4614e^{-02} \\ 6.7159e^{-02} & 4.4614e^{-02} & 8.4418e^{-02} \end{bmatrix}$$

The solution is clearly symmetric. To verify its positive definiteness, its eigenvalues were computed and found to be at $\{3.4331e^{-02}, 3.6532e^{-02}, 2.1466e^{-01}\}$.

Example 2: (Discrete Time Systems) Keeping the descriptor matrix E as the same, a state matrix was chosen such that the magnitude generalized eigenvalues of the pencil $(A - \lambda E)$ was less than unity. The system description is given by

$$\begin{bmatrix} 3 & 4 & 2 \\ 2 & 3 & 4 \\ 3 & 1 & 5 \end{bmatrix} \mathbf{x}(k+1) = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 2 & 2 \\ 2 & 1 & 0 \end{bmatrix} \mathbf{x}(k).$$

Keeping the EQE^T matrix the same as Example 1,

$$EQE^T = \begin{bmatrix} 2.2911e^{+00} & 1.7090e^{+00} & 1.1907e^{+00} \\ 1.7090e^{+00} & 1.3618e^{+00} & 8.9261e^{-01} \\ 1.1907e^{+00} & 8.9261e^{-01} & 9.5121e^{-01} \end{bmatrix},$$

the solution of the discrete time generalized Lyapunov equation was found to be

$$P = \begin{bmatrix} +3.6395e^{-01} & -3.8100e^{-01} & +1.6426e^{-01} \\ -3.8100e^{-01} & +8.1348e^{-01} & -3.4939e^{-01} \\ +1.6426e^{-01} & -3.4939e^{-01} & +1.9711e^{-01} \end{bmatrix}$$

Once again, the matrix is clearly symmetric and its eigenvalues are located at $\{3.9056e^{-02}$, $1.5383e^{-01}$, $1.1816e^{+00}\}$.

5. Concluding Remarks

In this paper, we attempted to provide numerical solution if matrix Lyapunov equations arising in the study of stability of generalized state space systems and the computation of their controllability and observability Gramians. Interestingly enough the results nicely parallel the case of standard state space systems. The results reported here are quite incomplete and further research in developing viable computational schemes is currently under way. Other topics of interest that follow from this study is to attempt a characterization of Lyapunov equation for singularly perturbed systems and (perhaps) singular systems.

References

[1] R.W. Brockett, *Finite Dimensional Linear Systems*, John Wiley, New York, 1970

[2] T. Kailath, *Linear Systems*, Prentice Hall, Englewood Cliffs, New Jersy, 1980.

[3] R.A. DeCarlo, *Linear Systems*, Prentice Hall, New York, 1989.

[4] D.F. Delchamps, *State Space and Input-Output Linear Systems*, Springer Verlag, New York, 1988.

[5] W.J. Rugh, *Linear System Theory*, Prentice Hall, Englewood Cliffs, New Jersy, 1992.

[6] R.H. Bartels and G.W. Stewart, "Solution of the Equation $AX + XB = C$," *Comm. Assoc. Com. Mach.*, vol. 15, pp. 820-826, 1972.

[7] G.H. Golub, S. Nash and C. VanLoan, "A Hessenberg-Schur method for the problem $AX + XB = C$," *IEEE Trans. Automat. Contr.*, vol. AC-24, pp. 901-913, 1979.

[8] S.J. Hammerling, "Numerical solution of the stable, nonnegative definite Lyapunov equation," *IMA Jour. Numer. Anal.*, vol. 2, pp. 303-323, 1982.

[9] J.W. Brewer, "Kronecker Products and Matrix Calculus in System Theory," *IEEE Trans. Cir. Syst.*, vol. CAS-25, pp. 772-781, 1978.

[10] G.W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[11] G.H. Golub and C. VanLoan, *Matrix Computations*, The John Hopkins Press, Baltimore, MD, 1983.

A Quadratically Convergent Local Algorithm on Minimizing the Largest Eigenvalue of a Symmetric Matrix

Michael K.H. Fan

School of Electrical Engineering
Georgia Institute of Technology, Atlanta, GA 30332

Abstract

Optimization problems involving eigenvalues arise in many engineering problems. In this paper, we continue the work reported in [1] and propose a new algorithm on minimizing the largest eigenvalue of a symmetric matrix which depends on the decision variable x affinely. Under certain mild assumptions, it is shown that, if started close enough to the minimizer x^*, the proposed algorithm will converge to x^* quadratically. The proposed algorithm has been extended on minimizing sums of the largest eigenvalues of a symmetric matrix [2].

0. Notation

I	Identity matrix of appropriate size
x^T	Transpose of vector x
$\|x\|$	Euclidean norm of vector x
A^T	Transpose of matrix A
$A \geq 0$	Matrix A is positive semi-definite
$A^\dagger$	Moore-Penrose generalized inverse of matrix A
$\mathrm{tr}(A)$	Trace of matrix A

Let $\eta(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^{k_1 \times k_2}$ for some positive integers m, k_1, and k_2. For any positive integer p, the notation $\eta(x) = O(\|x\|^p)$ means that $\|\eta(x)\|$ goes to zero at least as fast as $\|x\|^p$ does, where $\|\eta(x)\|$ denotes any matrix induced norm of $\eta(x)$. More precisely, it means that there exists $K \geq 0$ such that $\|\eta(x)\|/\|x\|^p \leq K$ as $x \rightarrow 0$.

1. Introduction

Optimization problems involving eigenvalues arise in many engineering problems. In this paper, we consider the problem on minimizing the largest eigenvalue of a symmetric matrix which depends affinely on the decision variable x. More precisely, given symmetric matrices A_i, $i = 0, \ldots, m$, we consider the convex optimization problem

$$\lambda^* = \inf_{x \in \mathbb{R}^m} \lambda_1(x) \tag{1.1}$$

where $\lambda_1(x), \ldots, \lambda_n(x)$ denote the eigenvalues, arranged in decreasing order, of

$$A(x) = A_0 + \sum_{i=1}^{m} x_i A_i \ .$$

See [1] and references therein for applications of (1.1) and existing methods for solving (1.1). In this paper, we continue the work reported in [1] and propose a new algorithm for solving (1.1). Under certain mild assumptions, it is shown that, if started close enough to the minimizer x^*, the proposed algorithm will converge to x^* quadratically. The algorithm has been extended on minimizing sums of the largest eigenvalues of a symmetric matrix [2].

Our method is based on the result that, in a neighborhood of x^*, (1.1) can be reformulated as a *smooth* constrained optimization problem [1]. Specifically, given $x \in \mathbb{R}^m$, the multiplicity of $\lambda_1(x)$ is said to be q if $\lambda_1(x) = \lambda_q(x) > \lambda_{q+1}(x)$. Suppose that (1.1) is achieved at some minimizer x^*. Let q be the multiplicity of $\lambda_1(x^*)$ and define

$$f_1(x) = \frac{1}{q} \sum_{i=1}^{q} \lambda_i(x)$$

and

$$f_2(x) = \frac{1}{2} \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} (\lambda_i(x) - \lambda_j(x))^2 \ .$$

Then, it is shown in [1] that $f_1(x)$ and $f_2(x)$ are analytic in x^*. Furthermore, it holds that

$$\lambda^* = \min_{x \in \mathbb{R}^m} \{ f_1(x) : f_2(x) = 0 \} \ . \tag{1.2}$$

Therefore, in the vicinity of x^*, one could apply smooth optimization techniques in solving (1.2). One of such techniques is the SQP method, which has superlinear rate of convergence in the nondegenerate case. However, as mentioned in [1], since the first order approximation of $f_2(x)$ at x^* is always degenerate, the superlinear rate of convergence using SQP cannot be expected. Therefore, special consideration is necessary in order to derive a fast algorithm. Our algorithm explores some special properties of the functions $f_1(x)$ and $f_2(x)$. Another quadratically convergent algorithm can be found in [3].

The organization of the paper is as follows. In Section 2, we give the assumptions with a brief discussion. A new optimality condition for (1.2) (or, equivalently, for (1.1)) is derived in Section 3. It will be used as the stopping criterion. The proposed algorithm with its convergence property is given in Section 4. In Section 5, we briefly discuss the computational complexity. Finally, an example is presented in Section 6 to illustrate the convergence property.

In the sequel, an eigenvalue decomposition of a symmetric matrix is referred to one with decreasing eigenvalues and orthonormal eigenvectors.

2. Assumptions

Suppose that (1.1) is achieved at a minimizer x^*. Recall that $f_1(x)$ and $f_2(x)$ defined in Section 1 are analytic at x^*. Therefore, when both $x - x^*$ and h are sufficiently small, for $i = 1, 2$, $f_i(x + h)$ is equal to its Taylor series expansion about x, i.e.,

$$f_i(x+h) = f_i(x) + g_i^T(x)h + \frac{1}{2}h^T H_i(x)h + O(\|h\|^3), \qquad i = 1,2$$

where $g_i(x)$ and $H_i(x)$ are the gradient and Hessian of $f_i(x)$, respectively. Expressions for $g_i(x)$ and $H_i(x)$ can be found in [1]. To show the convergence result, we will not need these expressions except the following properties they possess: (i) $g_2(x) = 0$ if $f_2(x) = 0$. (ii) $H_2(x) = H_{21}(x) + H_{22}(x)$; $H_{21}(x)$ and $H_{22}(x)$ are symmetric; $H_{21}(x) \geq 0$; $H_{22}(x) = 0$ if $f_2(x) = 0$.

Let the columns of $N(x)$ form an orthonormal basis for the null space of $H_{21}(x)$. Given $h \in \mathbb{R}^m$, define $D(x, h)$ to be the first order approximation of the directional derivative of $H_{21}(x)$ in the direction h, i.e., $D(x, h)$ is linear in h and satisfies

$$\lim_{t \to 0} \frac{H_{21}(x + th) - H_{21}(x)}{t} = D(x,h) + O(\|h\|^2) \ .$$

Also, let $\hat{D}(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^{m \times m}$ be such that $D(x,h)\phi = \hat{D}(x,\phi)h$ for all $x, h, \phi \in \mathbb{R}^m$. The following proposition gives the expressions of $D(x, h)$ and $\hat{D}(x, \phi)$. They can be verified using a result in [4].

Proposition 2.1. Let $x, h, \phi \in \mathbb{R}^m$. Then

$$D(x,h) = \sum_{k,l}^{q} \left((T_k - T_l)h(t_k - t_l)^T + (t_k - t_l)h^T(T_k - T_l) + 2q\left(T_{kl}ht_{kl}^T + t_{kl}h^T T_{kl}^T\right) \right)$$

$$\hat{D}(x,\phi) = \sum_{k,l}^{q} \left(\left((t_k - t_l)^T \phi I + (t_k - t_l)\phi^T\right)(T_k - T_l) + 2q\left(t_{kl}^T \phi I + t_{kl}\phi^T\right) T_{kl} \right)$$

where

$$t_k = t_{kk}$$

$$
\begin{aligned}
T_k &= T_{kk} \\
(t_{kl})_i &= u_k^T(x)A_i u_l(x) \\
(T_{kl})_{ij} &= u_k^T(x)A_i S_l A_j u_l(x) + u_l^T(x)A_i S_k A_j u_k(x) \\
S_k &= U_\perp(\lambda_k(x)I - \Lambda_\perp)^{-1}U_\perp^T \\
U_\perp &= [u_{q+1}(x) \ \cdots \ u_n(x)] \\
\Lambda_\perp &= \operatorname{diag}(\lambda_{q+1}(x), \cdots, \lambda_n(x)) \ .
\end{aligned}
$$

Here, given a vector t, we denote by $(t)_i$ its i-th element, and given a matrix T, we denote by $(T)_{ij}$ its ij-th element. Also, to simplify the notation, we denote $\sum_{k=1}^{q-1}\sum_{l=k+1}^{q}$ by $\sum_{k,l}^{q}$. □

We shall make the following assumptions for the data in (1.1):

Assumption 2.1. *(i)* (1.1) is achieved at some minimizer x^*. *(ii)* The multiplicity of $\lambda_1(x^*)$ is known. *(iii)* $N^T(x^*)Q(x^*)N(x^*)$ is nonsingular (the definition of $Q(x^*)$ will be given later). *(iv)* $H_{21}(x)$ has constant rank in a neighborhood of x^*. *(v)* The tangent plane of $f_2(x)$ at $x = x^*$ has dimension same as that of the null space of $H_{21}(x^*)$ (see, e.g., [5] for the definition of tangent plane). □

Let us briefly discuss the assumptions. The solution λ^* of (1.1) may be $-\infty$. However, if λ^* is finite, it must lie between the smallest and largest eigenvalues of A_0. Therefore, if the assumption *(i)* is not satisfied, one can check this easily in solving (1.1). The assumption *(ii)* is crucial in defining the functions $f_1(x)$ and $f_2(x)$. The multiplicity of $\lambda_1(x^*)$ can be identified when x is approaching to x^*. The assumption *(iii)* is closely related to the uniqueness of x^*. Together with other assumptions, we shall show in Section 3 that x^* is indeed unique. The assumption *(iv)* is given for a technical reason. We have tested many random examples and observed that this assumption seems to hold generically. In fact, it suffices to assume that the rank of $H_{21}(x^*)$ is known, which again can be identified when x is approaching x^*. The last assumption is related to a new optimality condition that we will propose in next section. Again, we have tested many random examples and observed that this assumption seems to hold generically as well.

3. Optimality conditions

We discuss optimality conditions for (1.1). Since $\lambda_1(x)$ is convex in x, from the theory of convex analysis, x^* solves (1.1) if and only if the origin is contained in the generalized gradient of $\lambda_1(x)$ at x^* (see, e.g., [6]). This condition can be more specific when the definition of $\lambda_1(x)$ is used. Let $A(x^*) = U(x^*)\Lambda(x^*)U^T(x^*)$ be an eigenvalue decomposition of $A(x^*)$, where $\Lambda(x^*) = \operatorname{diag}(\lambda_1(x^*), \cdots, \lambda_n(x^*))$ and $U(x^*) = [u_1(x^*) \ \cdots \ u_n(x^*)]$. Let q denote the multiplicity of $\lambda_1(x^*)$. Define $\Lambda_q(x^*) = \operatorname{diag}(\lambda_1(x^*), \cdots, \lambda_q(x^*))$ and $U_q(x^*) = [u_1(x^*) \ \cdots \ u_q(x^*)]$. Then x^* solves (1.1) if and only if there exists a $q \times q$ symmetric positive semi-definite matrix Q satisfying $\operatorname{tr}(Q) = 1$ and, for $i = 1, \cdots, m$, $\operatorname{tr}\left(U_q^T(x^*)A_iU_q(x^*)Q\right) = 0$ (see, e.g., [1,3]). In [1], optimality conditions are also derived for x to satisfy $\lambda_1(x) - \epsilon \leq \lambda^* \leq \lambda_1(x)$ for any given $\epsilon \geq 0$.

However, the above optimality conditions usually cannot be easily verified. In fact, the problem of determining whether the origin is contained in the generalized gradient of $\lambda_1(x)$ is similar to the optimization problem (1.1) (but with possibly much smaller n and m than those of the original problem). In Theorem 3.1 below, we propose a new optimality condition for x^* to solve (1.1). This condition can be easily checked and will be used as the stopping criterion for the algorithm that we are going to propose. It is noticed that the new optimality condition is merely shown to be necessary for x^* to satisfy (1.1) (see below). Therefore, we still need to check one of the (necessary and sufficient) optimality conditions mentioned above. However, this is only needed after the new optimality condition is satisfied so that reduction of the overall effort can be expected.

Theorem 3.1. Suppose that the assumptions given in Section 2 are satisfied at some minimizer x^* for (1.1). Then, the following statements hold. *(i)* x^* is the *unique* minimizer for (1.1). *(ii)* $N(x^*)$ is either a null matrix or one satisfying

$$N^T(x^*)g_1(x^*) = 0 \ .$$

(iii) There exists a neighborhood B of x^* such that, for any $x \in B$, $x \neq x^*$, x is either infeasible, i.e., $f_2(x) \neq 0$, or $N^T(x)g_1(x) \neq 0$. □

Theorem 3.1 implies that, in a neighborhood of x^*, x^* is the only point which is feasible (i.e., $f_2(x) = 0$) and satisfies the condition that $N(x)$ is either a null matrix or one satisfying $N^T(x)g_1(x) = 0$. Therefore, a natural choice of the stopping criterion is: *stop if* $|f_2(x^k)|$ *is sufficiently small and, if* $N(x^k)$ *is not null,* $\|N^T(x^k)g_1(x^k)\|$ *is sufficiently small.*

4. The algorithm

Assume that x is sufficiently close to x^*. The following algorithm computes a new estimate x_{new} from x, i.e., it performs one iteration.

Algorithm 4.1. Let $\phi_2 = -H_{21}^{\dagger}(x)g_2(x)$. Define $h(x) = \phi_2$ if $N(x)$ is a null matrix. Otherwise, let $\phi_1 = -H_{21}^{\dagger}(x)g_1(x)$, $Q(x) = H_1(x) + \hat{D}(x, \phi_1)$, and define

$$h(x) = \phi_2 - N(x)\left(N^T(x)Q(x)N(x)\right)^{-1} N^T(x)(Q(x)\phi_2 + g_1(x)) .$$

Then, x_{new} is defined by

$$x_{\text{new}} = x + h(x) . \qquad \square$$

Notice that, by Assumption 2.1 *(iii)* and continuity, $N^T(x)Q(x)N(x)$ is also nonsingular provided that x is sufficiently close to x^*.

The convergence property of the algorithm is summarized as follows.

Theorem 4.1. Let x_{new} be defined as above. Then $\|x_{\text{new}} - x^*\| = O\left(\|x - x^*\|^2\right)$. $\square$

Theorem 4.1 asserts that in a region near x^*, there exists some $M > 0$ such that $\|x^{k+1} - x^*\| \leq M\|x^k - x^*\|^2$. We see that if $M\|x^k - x^*\| < 1$, then $\|x^{k+1} - x^*\| < \|x^k - x^*\|$ and thus we conclude that if started close enough to the minimizer x^*, the algorithm will converge to x^* quadratically.

5. Computational complexity

We discuss in this section the computational complexity of the proposed algorithm. More specifically, we show the order of the number of flops needed for each iteration of the algorithm (i.e., to compute x_{new} from x). Each iteration requires an eigenvalue decomposition of an n by n symmetric matrix ($O(n^3)$ flops), computing gradients, Hessians, etc ($O(qmn^2)$ flops), and the various quantities defined in Algorithm 4.1 ($O(m^3)$ flops). Therefore, the overall computation effort for each iteration depends upon the size of problem polynomially.

6. An example

The algorithm outlined in Section 4 has been implemented in Matlab [7]. In this section, we illustrate its convergence property using an example in [1] ($n = m = 5$ and see [1] for the 6 symmetric matrices). First, we apply Kelley's cutting plane method [8] and

iteration k	$\lambda_1(x^k)$	$\|N^T(x^k)g_1(x^k)\|$	$\|x^k - x^*\|$
0	0.78	1.6×10^{-1}	2.2×10^{-1}
1	0.710	1.2×10^{-2}	1.0×10^{-1}
2	0.7094	7.7×10^{-4}	2.7×10^{-3}
3	0.7088827	3.2×10^{-6}	5.7×10^{-6}
4	0.708882596891	1.9×10^{-11}	6.5×10^{-11}
5	0.708882596889983	5.6×10^{-16}	5.8×10^{-15}

Table 6.1. Numerical result for running Algorithm 4.1 on an example in [1]. The second column is the objective function to be minimized. The third column $\|N^T(x^k)g_1(x^k)\| = 0$ together with the feasibility $f_2(x^k) = 0$ is used as the stopping criterion. The last column is the distance between x^k and the minimizer x^*.

---

obtain

$$x^* = \begin{bmatrix} -0.61362184194611 \\ 0.61455471338511 \\ -0.34375305063980 \\ -0.60674529250943 \\ 0.64646006292819 \end{bmatrix} \qquad \begin{bmatrix} \lambda_1(x^*) \\ \lambda_2(x^*) \\ \lambda_3(x^*) \\ \lambda_4(x^*) \\ \lambda_5(x^*) \end{bmatrix} = \begin{bmatrix} 0.70888259688998 \\ 0.70888259688998 \\ 0.06141090940867 \\ -0.60339788478773 \\ -1.45442436443040 \end{bmatrix}$$

Thus the multiplicity of $\lambda_1(x^*)$ is 2. We then choose the initial guess

$$x = x^* + [0.1\ 0.1\ 0.1\ 0.1\ 0.1]^T$$

and run the proposed algorithm. Appendix A contains a Matlab M-file which implements the proposed algorithm for the case $q = 2$. The M-file is used to carry out the numerical experiment and the result is summaried in Table 6.1. It is seen that, as a result of the quadratic rate of convergence, the proposed algorithm takes very few iterations to obtain the solution with accuracy close to the machine precision.

Appendix A

The following code is a Matlab M-file that implements Algorithm 4.1 when the multiplicity of $\lambda_1(x^*)$ is 2. In addition to the assumptions made in Section 2, here we also assume that $N(x^*)$ is not a null matrix. In the argument list below, `A0` is an $n^2 \times 1$ vector containing A_0, regarded as a single column. `AA` is an $n^2 \times m$ matrix; its i-th column is A_i, again, regarded as a single column. `x` is the initial guess (which is assumed to be sufficiently close to x^*). On return, `x` is the (unique) minimizer and `e` the corresponding eigenvalues. The code has not been optimized for speed.

```
function [x,e] = algorithm(n,m,A0,AA,x)

Ai  = zeros(n);
A   = zeros(n);
t1  = zeros(m,1);
t2  = zeros(m,1);
t12 = zeros(m,1);

while 1,
   A(:) = A0 + AA*x;

   [U,e] = eig(A);                    % eigenvalues and eigenvectors
   [e,K] = sort(real(diag(-e)));      % sort eigenvalues in decreasing order
   e = -e;
   U = U(:,K);
   u1 = U(:,1);                       % eigenvector corresponding to e(1)
   u2 = U(:,2);                       % eigenvector corresponding to e(2)

   Up = U(:,3:n);
   Lp = e(3:n);

   S1 = Up*diag(1 ./(e(1)-Lp))*Up';
   S2 = Up*diag(1 ./(e(2)-Lp))*Up';

   w1 = []; w2 = []; W1 = []; W2 = [];

   for i=1:m,
      Ai(:)   = AA(:,i);
      Aiu1    = Ai*u1;
      Aiu2    = Ai*u2;
      t1(i)   = u1'*Aiu1;
      t2(i)   = u2'*Aiu2;
      t12(i)  = u1'*Aiu2;
      w1      = [w1 Aiu1];
      w2      = [w2 Aiu2];
      W1      = [W1 S1*Aiu1];
      W2      = [W2 S2*Aiu2];
   end

   T1  = 2*w1'*W1;
   T2  = 2*w2'*W2;
   T12 = w1'*W2 + w2'*W1;

   g1 = t1 + t2;                      % gradient of f1
   H1 = T1 + T2;                      % Hessian  of f1

   g2  = (e(1)-e(2))*(t1-t2);         % gradient of f2
   H21 = (t1-t2)*(t1-t2)'+4*t12*t12';

   N = null(H21);                     % null  space of H21
   R = orth(H21);                     % range space of H21
   if norm(N'*g1) < 1e-12,             % check optimality condition
      break;
   end

   H = R*((R'*H21*R)\R');              % generalized inverse of H21
   phi1 = -H*g1;
   phi2 = -H*g2;
   Dh = ((t1-t2)'*phi1*eye(m)+(t1-t2)*phi1')*(T1-T2) + ...
        4*(t12'*phi1*eye(m)+t12*phi1')*T12;
   Q = H1 + Dh;
   h = phi2 - N*((N'*Q*N)\(N'*(Q*phi2+g1)));
   x = x + h;
end
```

References

[1] M.K.H. Fan & B. Nekooie, "On minimizing the largest eigenvalue of a symmetric matrix," *Proceedings of 31th IEEE Conference on Decision and Control*, Tucson, Arizona (December 1992).

[2] B. Nekooie & M.K.H. Fan, "A quadratically convergent local algorithm on minimizing sums of the largest eigenvalues of a symmetric matrix," *Proceedings of 31th IEEE Conference on Decision and Control*, Tucson, Arizona (December 1992).

[3] M.L. Overton, "On minimizing the maximum eigenvalue of a symmetric matrix," *SIAM J. Matrix Anal. Appl.* 9 (1988).

[4] M.K.H. Fan, N.K. Tsing & E.I. Verriest, "On partial derivatives of functions involving eigenvalues," *summited to Linear Algebra and Its Applications.*

[5] D.G. Luenberger, *Linear and Nonlinear Programming*, Addison Wesley, 1989.

[6] F.H. Clarke, *Optimization and Nonsmooth Analysis*, J. Wiley & Sons, 1983.

[7] C. Moler, J. Little & S. Bangert, *PRO-MATLAB user's guide*, The MathWorks, Inc., Sherborn, MA, 1987.

[8] J.E. Kelley, "The cutting-plane method for solving convex programs," *J. Soc. Indust. Appl. Math.* 8 (December 1960), 703–712.

Iterative Image Restoration using FFT–Based Preconditioners

James G. Nagy [*] and Robert J. Plemmons [†]

September 25, 1992

Abstract

Image restoration can be modeled as a discrete ill-posed inverse problem where, if the blurring function (point spread function) is spatially invariant, then the blurring operator will have a block Toeplitz structure. We consider solving this discrete ill-posed image restoration problem by the preconditioned conjugate gradient algorithm. Because the problem is ill-posed, it is necessary to use regularization in order to compute a reasonable approximation to the true image. We compare preconditioning techniques for two different regularization procedures; Tikhonov regularization, and iterative regularization with the conjugate gradient algorithm. In particular, we show how one can incorporate preconditioning, using a block circulant with circulant blocks matrix, into the conjugate gradient algorithm for these regularization schemes. We thus obtain fast 2-D FFT–based iterative methods in which preconditioning takes place in the Fourier domain, while the iteration takes place in the spatial domain.

Key Words. Image restoration, Toeplitz-block system, FFT–based preconditioners, ill-posed inverse problems, conjugate gradients, least squares, regularization, deconvolution.

AMS(MOS) Subject Classifications. 65F10, 65F15, 43E10.

1 Introduction

Image restoration problems (*i.e.,* problems in which one wishes to remove or minimize degradations in a blurred and noisy image) occur in a variety of important applications such as remote sensing, medical imaging and space activities. Given an $n \times n$ image of pixels, the image restoration problem is typically modeled as the discrete ill–posed problem

$$g = Hf + \eta \tag{1}$$

where the n^2–vectors f, g and η represent the known image, the observed image and additive noise, respectfully. The $n^2 \times n^2$ matrix H represents the blur in the image, and

---

[*] Department of Mathematics, Southern Methodist University, Dallas, TX 75275-0156.

[†] Department of Mathematics and Computer Science, Wake Forest University, Box 7388, Winston-Salem, NC 27109. Research supported by the US Air Force under grant no. AFOSR–91–0163 and NSF grant no. CCR–92–01105.

will have a block Toeplitz structure if the degradation phenomena is spatially invariant. An $n \times n$ matrix T is called a Toeplitz matrix if the entries of T satisfy $t_{ij} = t_{i-j}$, i.e., the entries are constant down each diagonal. If the entries of an $n \times n$ Toeplitz matrix C also satisfy $c_{n-j} = c_{-j}$, then C is also called a *circulant* matrix.

As mentioned above, if the blurring function (point spread function) is spatially invariant, then the matrix H will have a Toeplitz structure. In particular, H will be an $n \times n$ block Toeplitz matrix, with each block being an $n \times n$ (point) Toeplitz matrix. That is,

$$H = \begin{bmatrix} T_0 & T_{-1} & \cdots & T_{-n+1} \\ T_1 & T_0 & \cdots & T_{-n+2} \\ \vdots & \vdots & & \vdots \\ T_{n-1} & T_{n-2} & \cdots & T_0 \end{bmatrix} \tag{2}$$

where each block T_k is an $n \times n$ point Toeplitz matrix. We call a matrix of this form a *block Toeplitz with Toeplitz block matrix*, and denote it by BTTB(n).

It is important to note that in image restoration the dimensions of the matrices can become quite large. For example, in low resolution imagery, the observed image might be discretized into 256×256 pixels, resulting in a matrix H which is $65,386 \times 65,386$. In high resolution imagery, the number of pixels is on the order of 10^3, which implies H will have dimensions on the order of $10^6 \times 10^6$ [14]. These sizes make the use of iterative methods, such as conjugate gradients, almost essential.

Another important aspect of image restoration is that it is modeled as a discrete ill–posed inverse problem. This means the matrix H can be very ill–conditioned, and hence the solution f will be extremely sensitive to perturbations in the right hand side. Thus, the presence of the noise term η makes this a nontrivial problem to solve. *Regularization* is often used to alleviate the sensitivity of (1) to the noise. Regularization has the effect of filtering out the noise at the expense of restricting the set of admissible solutions.

In this paper we consider two regularization schemes, and use the preconditioned conjugate gradient algorithm (PCG) to solve the resulting systems. The conjugate gradient (CG) method is an iterative method of fundamental importance for solving systems of linear equations $Hf = g$, *cf.*, Golub and van Loan [8]. The classical CG method applies only to the case where H is symmetric positive definite. But if H is not symmetric positive definite one can still use the CG method; for example, by considering the factored form of the normal equations

$$H^*(\mathbf{g} - H\mathbf{f}) = \mathbf{0},$$

where $*$ denotes the conjugate transpose. This variant of CG is generally called CGNR.

The convergence rate of the conjugate gradient algorithm depends on the singular values of the coefficient matrix H. If the singular values cluster around a fixed point, convergence will be rapid, *cf.* van der Sluis and van der Vorst [17]. Thus, to make the algorithm a useful iterative method, one usually *preconditions* the system. The preconditioned conjugate gradient algorithm, with preconditioner C, uses the conjugate gradient method to solve

$$HC^{-1}\mathbf{y} = \mathbf{g},$$

with $C\mathbf{f} = \mathbf{y}$.

This leads to the preconditioned conjugate gradient (PCGNR) algorithm for general linear systems and for least squares problems. The version of the PCGNR algorithm we use is given in [2] and can be stated as follows:

Algorithm 1. PCGNR. *Let* $\mathbf{f}^{(0)}$ *be an initial vector, and let* C *be a given preconditioner. This algorithm computes the solution,* $\mathbf{f}$*, to* $H\mathbf{f} = \mathbf{g}$*.*

```
• $\mathbf{r}^{(0)} = \mathbf{g} - H\mathbf{f}^{(0)}$
• $\mathbf{p}^{(0)} = \mathbf{s}^{(0)} = C^{-*}H^{*}\mathbf{r}^{(0)}$
• $\gamma_0 = \|\mathbf{s}^{(0)}\|_2^2$
• for $k = 0, 1, 2, \ldots$
    $\mathbf{q}^{(k)} = HC^{-1}\mathbf{p}^{(k)}$
    $\alpha_k = \gamma_k / \|\mathbf{q}^{(k)}\|_2^2$
    $\mathbf{f}^{(k+1)} = \mathbf{f}^{(k)} + \alpha_k C^{-1}\mathbf{p}^{(k)}$
    $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{q}^{(k)}$
    $\mathbf{s}^{(k+1)} = C^{-*}H^{*}\mathbf{r}^{(k+1)}$
    $\gamma_{k+1} = \|\mathbf{s}^{(k+1)}\|_2^2$
    $\beta_k = \gamma_{k+1} / \gamma_k$
    $\mathbf{p}^{(k+1)} = \mathbf{s}^{(k+1)} + \beta_k \mathbf{p}^{(k)}$
```

The preconditioners we use for solving (1) are block circulant with circulant blocks (BCCB) matrices, and were first introduced in R. Chan, J. Nagy and R. Plemmons [4] and M. Hanke, J. Nagy and R. Plemmons [10]. Similar preconditioners for well conditioned problems have been considered in R. Chan and X. Jin [3], in T. Chan and J. Olkin [6] and in Ku and J. Kuo [15]. Our preconditioners take advantage of the Toeplitz structure as well as being able to incorporate regularization into their construction. Here we compare these two schemes, and illustrate their effectiveness in image restoration computations.

This paper is outlined as follows. In Section 2 we give a brief discussion on Tikhonov regularization and conjugate gradients as an iterative regularization method. In Section 3, we consider preconditioning the BTTB matrix with a BCCB matrix, and indicate in Section 4 how these preconditioners need to be modified when regularization in required. In Section 5 we provide some numerical results and concluding remarks.

2 Regularized Computations

In this section we give a brief discussion on Tikhonov regularization and also on conjugate gradients as an iterative regularization method. The presentation here concentrates on some of the algebraic properties and on some computational considerations of these regularizaiton schemes. For more thorough treatments of these topics we refer the reader to [9,11,16] and the references therein.

Image restoration is often modeled as a discrete ill–posed problem $g = Hf + \eta$, where the vector g and the matrix H are known, and we want to compute an accurate approximation to f. Since the problem is ill–posed, the matrix H is severely ill–conditioned. Thus, naively solving $Hf = g$ will lead to extreme instability with respect to perturbations in g. Regularization methods are used to achieve stability for these problems.

In classical *Tikhonov regularization* [9], stability is attained by introducing a stability operator (called a regularization operator) which restricts the set of admissible solutions. Since this causes the solution to be biased, a scalar (called a regularization parameter) is introduced to control the degree of bias. More specifically, the regularized solution is computed as

$$\min \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} H \\ \alpha L \end{bmatrix} f(\alpha) \right\|_2 , \tag{3}$$

where α is the regularization parameter and the matrix L is the regularization operator.

The regularization operator L is usually chosen to be the identity matrix or a discretization of a 2-D differentiation operator [1,7]. Thus, L will also have a BTTB structure, and this assumption is used to construct a BCCB preconditioner for

$$\tilde{H} = \begin{bmatrix} H \\ \alpha L \end{bmatrix}.$$

In particular, if L is chosen to be the identity matrix, then our BCCB preconditioner $\tilde{C}$ for $\tilde{H}$ can be obtained by simply adding α to each eigenvalue of the BCCB preconditioner C for H. This implies, moreover, that $\tilde{C}$ is nonsingular and well–conditioned. We will expand on these details in the next section.

In iterative regularization, such as CGNR, one applies the conjugate gradient iterations to the normal equations of the original system,

$$H^*Hf = H^*g.$$

At the beginning of the iteration process, the original signal f begins to be reconstructed. As the iterations increase, the noise contributions start to affect the minimization process [11,16]. At this point, the CGNR starts to reconstruct noise, rather than useful information. It is at this point, where we have reconstructed as much useful information as possible and the noise begins to dominate, that we stop the process. Thus, the regularization parameter here is the iteration index.

Preconditioning can be used to speed up the convergence of the CGNR, but care has to be taken if it is being used as a regularization process. Specifically, a good preconditioner should approximate the matrix H. Intuitively, if H is ill–conditioned, then we would expect a good approximation to H to also be ill–conditioned. Since each iteration requires solving a system with the preconditioner as the data matrix, an ill-conditioned preconditioner will cause the noise to dominate even the first few iterations, and hence we will not be able to recover the solution. We discuss in the next section how we can construct an effective, well–conditioned preconditioner for an ill–conditioned H.

Choosing α in Tikhonov regularization, as well as the stopping iteration index in iterative regularization, is a non trivial problem. It is not the purpose of this paper to focus on the specifics of these regularization procedures. Rather, we concentrate on the preconditioning techniques for speeding convergence to the desired solution.

3 Preconditioning Methods

We first consider the T. Chan [5] circulant approximation for a square point Toeplitz matrix. His circulant approximation is optimal with respect to the Frobenius norm, is defined for arbitrary square matrices A, and is constructed by averaging the entries along certain pairs of diagonals of A. In particular, for a given generic $n \times n$ matrix A, let $\mathcal{C}(A)$ be the $n \times n$ circulant approximation of A as defined in T. Chan [5], *i.e.*, C is the minimizer of $F(X) = ||A - X||_F$ over all circulant matrices X. For the special case where $A \equiv T$, and T is Toeplitz, we have

$$C = \mathcal{C}(A) = (c_{j-\ell})_{j,\ell},$$

where the (j,ℓ)th entry of C is given by the diagonal

$$c_k = \begin{cases} \dfrac{(n-k)t_k + kt_{k-n}}{n} & 0 \le k < n, \\ c_{n+k} & 0 < -k < n. \end{cases} \tag{4}$$

Throughout this paper we will use the T. Chan approximation $\mathcal{C}(\cdot)$ exclusively as a basic tool to construct our preconditioners for BTTB problems.

We now describe a BCCB preconditioner for BTTB, *i.e.*, 2-D, problems. It is based on second level approximations and will be called the *Level–2 preconditioner.* Our Level–2 preconditioner is based on circulant approximations at the Toeplitz-block level and again at a second, block Toeplitz level. The BTTB matrix H in (2) is approximated by a second-level circulant preconditioner, denoted by $\mathcal{C}_2(H)$. The preconditioner is related to the one proposed by T. Chan and Oklin [6] for square Toeplitz-block matrices and extended to least squares problems by R. Chan, Nagy and Plemmons [4].

For the BTTB matrix H given by (2), we define a first-level circulant approximation by

$$\mathcal{C}_1(H) \equiv \begin{bmatrix} \mathcal{C}(T_0) & \mathcal{C}(T_{-1}) & \cdots & \mathcal{C}(T_{-n+1}) \\ \mathcal{C}(T_1) & \mathcal{C}(T_0) & \cdots & \mathcal{C}(T_{-n+2}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C}(T_{n-1}) & \mathcal{C}(T_{n-2}) & \cdots & \mathcal{C}(T_0) \end{bmatrix}. \tag{5}$$

Now we apply another first-level circulant approximation $\tilde{\mathcal{C}}_1$ to $\mathcal{C}_1(H)$ to get the second-level circulant approximation $\mathcal{C}_2(H)$ that we want. The matrix $\tilde{\mathcal{C}}_1\{\mathcal{C}_1(H)\}$ is obtained by treating each block $\mathcal{C}(T_\mu)$ in $\mathcal{C}_1(H)$ as an entry of a matrix and then applying the formula (4) to this "point" matrix. More precisely, we form the block-circulant matrix

$$\tilde{\mathcal{C}}_1\{\mathcal{C}_1(H)\} = \begin{bmatrix} \tilde{C}_0 & \tilde{C}_{-1} & \cdots & \tilde{C}_{-n+1} \\ \tilde{C}_1 & \tilde{C}_0 & \cdots & \tilde{C}_{-n+2} \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{C}_{n-1} & \tilde{C}_{n-2} & \cdots & \tilde{C}_0 \end{bmatrix}$$

where the $\tilde{C}_\mu$ are computed using (4). Since each $\mathcal{C}(T_\mu)$ is circulant, we see that the sums $\tilde{C}_\mu$ are circulant matrices. Moreover, by their definition, we see that $\tilde{C}_\mu = \tilde{C}_{\mu-n}$, $0 < \mu < n$. In particular, we see that $\tilde{\mathcal{C}}_1\{\mathcal{C}_1(H)\}$ is then a block-circulant matrix with circulant blocks of the form BCCB(n). We note further that $\mathcal{C}_2(H) = \tilde{\mathcal{C}}_1\{\mathcal{C}_1(H)\}$, as given in Chan and Jin [3, Theorem 3]. It is shown in [3] that $\mathcal{C}_2(H)$ is "optimal" in the sense that it minimizes $\|H - X\|_F$ over all BCCB matrices X. For our purposes, we need only find the first column of $\mathcal{C}_1(H)$, and then use that column to find the first column of the final BCCB matrix $\mathcal{C}_2(H)$.

We note that the eigenvalues of $\mathcal{C}_2(H)$ can be obtained by taking the 2-D FFT of the first column of $\mathcal{C}_2(H)$. Specifically, if $\mathbf{c}_2$ denotes the first column of $\mathcal{C}_2(H)$ and `fft2d` denotes the 2-D FFT operator, then the vector of eigenvalues of $\mathcal{C}_2(H)$ is given by `fft2d`($\mathbf{c}_2$).

The construction of our Level–2 BCCB preconditioner (*i.e.*, the eigenvalues of $C \equiv \mathcal{C}_2(H)$) for the matrix H in (2) is summarized in the following algorithm:

Algorithm 2. Construction of the circulant **Level–2** preconditioner for 2-D problems. *Let H be the BTTB(n) matrix given as in* (2). *This algorithm computes the eigenvalues of the BCCB preconditioner C, using the 2-D FFT, and stores them in the diagonal matrix Λ.*

```
Find the first column, $\mathbf{c}_1$, of $C_1(H)$, using (4),
Find the first column, $\mathbf{c}_2$, of $C \equiv C_2(H) = \tilde{C}_1\{C_1(H)\}$ , using (4),
Find $\Lambda$ from $\Lambda 1 = \texttt{fft2d}(\mathbf{c}_2)$, using the 2-D FFT.
```

Since H is $n^2 \times n^2$, the cost of forming the Level–2 BCCB preconditioner C (*i.e.*, its eigenvalues in Λ), for the case where the matrix H is BTTB(n) , is $O(n^2(2\log n))$ operations. For implementation in the PCGNR algorithm, we need only the eigenvalues of C together with the 2-D FFT. We observe that for any vector $\mathbf{y}$, $C^{-1}\mathbf{y}$ can be computed by the formula

$$C^{-1}\mathbf{y} = \texttt{ifft2d}[\Lambda^{-1}\texttt{fft2d}(\mathbf{y})]$$

in

$$O(n^2(2\log n)) \tag{6}$$

operations by using `fft2d` and its inverse, `ifft2d`. FFT-based matrix-vector multiplications involving the BTTB matrices H and H^* in the PCGNR algorithm also have complexity (6), so (6) is the complexity per iteration of the PCGNR algorithm, implemented using the 2-D FFT. Recall that H is $n^2 \times n^2$.

We note that a convergence analysis of the PCGNR method with the BCCB preconditioner C is given in [4]. There it is shown that, for certain important classes of H, the singular values of the preconditioned matrix are clustered around one, leading to rapid convergence. It also follows from the analysis in [4] that the PCGNR iterations may be ineffective or even break down if H is very ill-conditioned. In the next section we review the results from [4] and [10] which show how regularization and preconditioning can be effectively combined by using the FFT–based spectral decomposition of C. The result is fast 2-D FFT–based algorithms for solving (1).

4 Regularization and Preconditioning

In this section we show how preconditioning can be incorporated into the PCG algorithm for computing regularized solutions to the discrete ill-posed problem (1). We first begin with Tikhonov regularization (3). To simplify the discussion, we assume that the regularization operator is the identity matrix. The preconditioning techniques we present here can be applied, with slight modifications, when $L \neq I$. We refer the reader to [4] for the specific details.

We now want to solve the least squares problem

$$\min \left\| \begin{bmatrix} g \\ 0 \end{bmatrix} - \begin{bmatrix} H \\ \alpha I \end{bmatrix} f(\alpha) \right\|_2 , \tag{7}$$

where H is a BTTB(n) matrix, using the PCGNR algorithm. Our goal is to construct a BCCB(n) matrix C_α so that the singular values of

$$\tilde{H} = \begin{bmatrix} H \\ \alpha I \end{bmatrix} C_\alpha^{-1}$$

are clustered around one. If we can do this, then the convergence of the PCGNR will be rapid [17].

Let C_H be the BCCB approximation to H as given by Algorithm 1. Then $H^*H + \alpha^2 I \approx C_H^* C_H + \alpha^2 I$. Because C_H is a BCCB matrix, it can be written as $C_H = \mathcal{F}^* \Lambda \mathcal{F}$, where $\mathcal{F}$ is the unitary 2-D Fourier transform matrix and Λ is a diagonal matrix containing the eigenvalues of C_H. Thus

$$H^*H + \alpha^2 I \approx \mathcal{F}^*[|\Lambda|^2 + \alpha^2 I]\mathcal{F} = \mathcal{F}^* D \mathcal{F}$$

where D is a diagonal matrix with (i,i) entry given by $|\lambda_i|^2 + \alpha^2$. Thus, we can write

$$H^*H + \alpha^2 I \approx \mathcal{F}^* D^{1/2} \mathcal{F} \mathcal{F}^* D^{1/2} \mathcal{F}$$

and we take C_α to be the BCCB matrix

$$C_\alpha = \mathcal{F}^* D^{1/2} \mathcal{F}. \tag{8}$$

Using this BCCB matrix C_α, we then solve (7) with the PCGNR algorithm. It is shown in [4] that for a wide class of BTTB matrices H, the PCGNR converges in $O(n)$ steps for n sufficiently large. Thus, the total cost for solving (7) for these systems is $O(n^3 \log n)$, where $\tilde{H}$ is $2n^2 \times n^2$.

We now turn to the problem of preconditioning the iterative regularization method of CGNR. Here one does not have a stabilizing operator as in Tikhonov regularization. Thus, to speed convergence of the CGNR, we must precondition H directly. As mentioned earlier, if this is done without much thought, then the preconditioner itself will be ill-conditioned, and hence we will be unable to recover an accurate approximation to f.

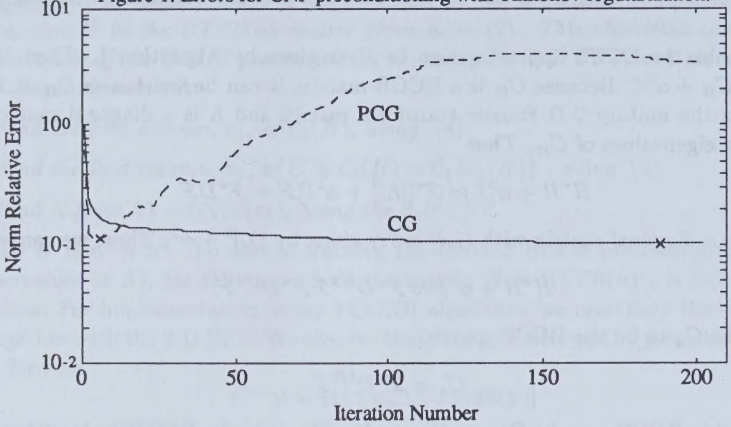
The effectiveness of the preconditioner depends on how well the spectrum of HC^{-1} is clustered. Intuitively, we would like to cluster as much of the spectrum of HC^{-1} as possible, while keeping C from becoming too ill-conditioned. To achieve this goal we first construct the BCCB approximation C_H to H as in Section 3. The conditioning of C_H is reflected in its eigenvalues, which can be obtained easily using the 2-D FFT. In [10], it is proposed to use a truncated spectral decomposition of C_H in order to construct a well conditioned C. Specifically, a *truncation parameter* τ is chosen, and the eigenvalues of C_H with magnitudes less than τ are replaced by 1. (The parameter τ depends on the signal and noise subspaces, and we refer the reader to [10] for further details on how to choose it.) This modified preconditioner, which we denote as C_τ, has the property that it clusters the spectrum on the signal subspace, while leaving the spectrum associated with the noise subspace unchanged. It is also shown in [10] that for a wide class of BTTB matrices, the PCGNR will give the best regularized solution to f in at most $O(n)$ steps for n sufficiently large.

5 Numerical Experiments

In this section we provide some numerical experiments to illustrate and compare the two preconditioned regularization methods discussed in this paper. The model problem we use arises in image restoration computations [13], with the BTTB matrix H given by

$$h_{i-j,k-l} = \begin{cases} \left[\frac{\sin((i-j)/\beta)}{(i-j)/\beta}\right]^2 \left[\frac{\sin((k-l)/\beta)}{(k-l)/\beta}\right]^2 & i-j, k-l \le \omega \\ 0 & \text{otherwise.} \end{cases}$$

Figure 1. Errors for CNP preconditioning with Tikhonov regularization.



	CNP Tikhonov Reg.		HNP Iterative Reg.	
	Prec.$=C_\alpha$	No prec.	Prec.$=C_\tau$	No prec.
iterations	6	188	19	148
min. error	0.1157	0.1046	0.1122	0.1047

Table 1: Summary of numerical results.

Here we take $n = 64$ and $\beta = 4$, which results in a 4096×4096 matrix with condition number on the order of 10^6. We construct a known simulated 64×64 image, and form the 4096 vector f by row ordering, as in [14]. The degraded image is constructed by forming the vector $g = Hf + \eta$. The entries of the noise vector η are generated randomly from a normal distribution with mean of 0.0 and variance 1.0, and scaled so that $\|\eta\|_2/\|Hf\|_2 = 10^{-3}$. To test the effectiveness of the preconditioners, we apply the CGNR with and without a preconditioner to both Tikhonov and iterative regularization.

In Figure 1 we plot the relative errors for several iterations of the CGNR applied to the Tikhonov regularized problem using the BCCB preconditioner C_α and no preconditioner. Here $\alpha = 0.10$, and was chosen based on several runs in order to obtain a near optimal regularization parameter. We can see from this plot that the preconditioner significantly improves the speed of convergence of the CGNR.

In Figure 2 we plot the relative errors for several iterations of the CGNR without a preconditioner, and with the preconditioner C_τ. Here τ was chosen, using a scheme proposed in [10], to be 0.03. Figure 4 also shows that the convergence is significantly faster when the preconditioner C_τ is used.

In Table 1, we summarize the number of iterations each required to reach its best regularized solution, along with the relative errors attained by each. We see that both regularization schemes are effective in stabilizing the computations so that a meaningful solution can be computed. Moreover, the results in Table 1 illustrate that both preconditioning schemes can be very effective in speeding the convergence to a solution.

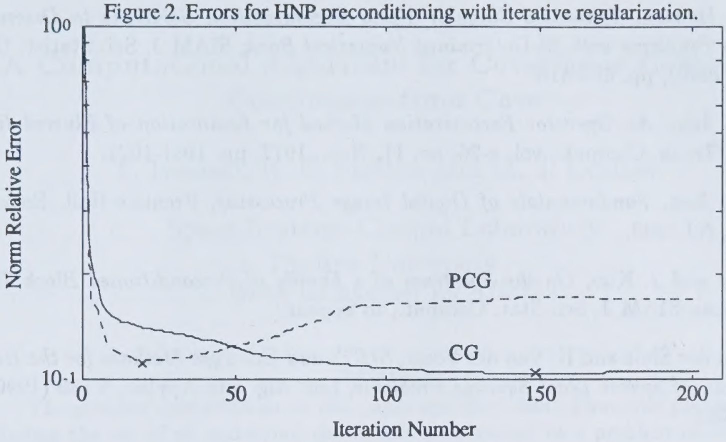


Figure 2. Errors for HNP preconditioning with iterative regularization.

References

[1] J. Biedmond, R. Lagendijk and R. Mersereau, *Iterative Methods for Image Deblurring,* Proc. of the IEEE, V78 (1990), pp. 856–883.

[2] A. Bjorck, *Least Squares Methods*, in Handbook of Numerical Methods, ed. P. Ciarlet and J. Lions, Elsevier/North Holland Vol. 1, 1989.

[3] R. Chan, X. Jin, *A Family of Block Preconditioners for Block Systems*, SIAM J. Sci. Statist. Comput., to appear.

[4] R. Chan, J. Nagy and R. Plemmons, *FFT-Based Preconditioners for Toeplitz-Block Least Squares Problems*, University of Minnesota IMA Technical Report # 957, April, 1992. To appear in SIAM J. Numer. Anal..

[5] T. Chan, *An Optimal Circulant Preconditioner for Toeplitz Systems*, SIAM J. Sci. Stat. Comp., V9 (1988), pp. 766–771.

[6] T. Chan and J. Olkin, *Preconditioners for Toeplitz–block matrices*, preprint, (1992).

[7] L. Eldén, *An Algorithm for the Regularization of Ill-conditioned, Banded Least Squares Problems*, SIAM J. Sci. Statist. Computing, V5 (1984), pp. 237-254.

[8] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, Baltimore, MD, 2nd Ed., 1989.

[9] C. Groetsch, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pittman Publishing, Boston, 1984.

[10] M. Hanke, J. Nagy and R. Plemmons *Preconditioned Iterative Regularization Methods,* preprint (1992).

[11] M. Hanke and P. C. Hansen, *Regularization Methods for Large-Scale Problems*, in preparation.

[12] P. C. Hansen, *Truncated Singular Value Decomposition Solutions to Discrete Ill-Posed Problems with Ill-Determined Numerical Rank*, SIAM J. Sci. Statist. Comp., V11 (1990), pp. 503-518.

[13] A. K. Jain, *An Operator Factorization Method for Restoration of Blurred Images*, IEEE Trans. Comput., vol. c-26, no. 11, Nov., 1977, pp. 1061-1071.

[14] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Engelwood Cliffs, NJ 1989.

[15] T. Ku and J. Kuo, *On the Spectrum of a Family of Preconditioned Block Toeplitz Matrices*, SIAM J. Sci. Stat. Comput., to appear.

[16] A. van der Sluis and H. Van der Vorst, *SIRT- and CG-Type Methods for the Iterative solution of Sparse Least Squares Problems*, Lin. Alg. and Applic., V130 (1990), pp. 257–302.

[17] A. van der Sluis and H. Van der Vorst, *The Rate of Convergence of Conjugate Gradients*, Numer. Math., V48 (1986), pp. 543–560.

A Computational Algorithm for Covariance Control : Continuous-time Case

T. Iwasaki, R. E. Skelton and M. J. Corless

Space Systems Control Laboratory
Purdue University
West Lafayette, IN 47907

Abstract

The primary contributions of this paper are three fold. First, the problem of finding the set of all stabilizing controllers is converted to a problem of finding a matrix which can be assigned to the closed loop system as a state covariance. Necessary and sufficient conditions for a given matrix to be assignable are given. Secondly, a finite step algorithm for constructing the set of all assignable covariances is presented. Finally, a controller design method which achieves the resulting closed loop plant state covariance "closest" to a given desirable covariance is given.

1 Introduction

In practical controller design problems, it is frequently required to meet the performance specification described by inequality constraints on the root mean square (RMS) value of *each* output. Covariance control theory [2, 7, 8, 9, 10, 3], was originally motivated by such multiple design objectives: the multiple inequality constraints can be satisfied by assigning a matrix, which carries the specified RMS value of each state on the diagonal, to the closed loop system as the state covariance.

Works on the development of covariance control theory so far have mainly focused on the characterization of the set of all assignable covariances and the explicit formula for obtaining the controller which assigns a specified covariance. These results suggest the following controller design procedure:

1) Find a desirable covariance which carries the specified closed loop system properties such as RMS values of states, robustness, etc.

2) Using the assignability conditions, check if there exists a controller (using *noise-free* measurements) which yields the specified covariance for closed loop system.

3) If it is assignable, then compute the controller using the explicit formula in terms of the covariance.

One difficulty arises from step 2) to 3) when the desirable covariance is not assignable, and this is almost always the case in practice. The old results do not suggest any method to find a *desirable and assignable* covariance. The second difficulty arises from the fact that the existing theory does not treat noisy measurements.

The main purpose of this paper is to propose a way to overcome these difficulties. First we complete the theory of covariance control by allowing measurement noise. Next, we present a solution to the following optimization problem:

Given a desirable covariance which is not necessarily assignable, find the assignable covariance "closest" to the desired one.

The precise problem formulation will be given in section 5. To this end, the covariance control problem is stated first in section 2, then, in section 3, we will present a characterization of the set of all closed loop state (consists of the plant and the controller states) covariances which a linear time-invariant continuous-time system may possess. All controllers which assign a specified assignable covariance are also given explicitly. This is a new result obtained for the system with the measurement noise as well as the process noise (the old results consider the process noise only). Next, the set of all assignable *plant* state covariances is characterized since the output performance of the closed loop system (such as RMS value of each output) can be specified by the plant state covariance only. Now, the controller design problem can be converted into the problem of finding a plant state covariance which carries the desired closed loop performance and also satisfies the assignability conditions. Therefore, it is essential to know how to find such a covariance. In section 4, we will propose a finite step algorithm for constructing the set of all assignable plant state covariances. Then finally, a solution to the aforementioned optimization problem will be given in section 5. Conclusions are given in section 6.

We will use the following notations. $\mathcal{E}[\cdot]$ denotes the expectation operator. For a matrix A, A^T denotes the transpose of A. A^+ is the unique Moore-Penrose inverse of A. $\|A\|_F$ is the Frobenius norm of A. $tr[A]$ is the trace of A. rank$[A]$ is the rank of A. SVD stands for the singular value decomposition. Most of the proofs for the theorems are omitted due to the limited pages.

2 Problem Formulation

We consider the following linear time-invariant continuous-time system (Σ_p)

$$\dot{x}_p = A_p x_p + B_p u + D_p w_p \tag{2.1}$$

$$z = M_p x_p + v \tag{2.2}$$

where $w_p \in \Re^{n_w}$ and $v \in \Re^{n_z}$ are zero mean white noise stochastic processes with covariances $W_p > 0$ and $V > 0$, respectively, and correlation $\mathcal{E}[w_p(t)v^T(\tau)] = W_{12}\delta(t-\tau)$. $x_p \in \Re^{n_p}$ is the plant state, $u \in \Re^{n_u}$ is the control input, and $z \in \Re^{n_z}$ is the available measurement. Throughout the paper, we will assume that (A_p,D_p) is controllable, (A_p,B_p) is stabilizable, and (A_p,M_p) is detectable. Consider a dynamic controller (Σ_c) of order n_c

$$\dot{x}_c = A_c x_c + B_c z \tag{2.3}$$

$$u = C_c x_c + D_c z \tag{2.4}$$

where $x_c \in \Re^{n_c}$ is the controller state. Combining (Σ_c) and (Σ_p), the closed loop system (Σ_{cl}) can be described by

$$\dot{x} = (A + BGM)x + (D + BGJ)w \tag{2.5}$$

where

$$x := \begin{bmatrix} x_p \\ x_c \end{bmatrix}, \quad w := \begin{bmatrix} w_p \\ v \end{bmatrix}, \quad A := \begin{bmatrix} A_p & 0 \\ 0 & 0 \end{bmatrix}, \quad B := \begin{bmatrix} B_p & 0 \\ 0 & I_{n_c} \end{bmatrix},$$

$$D := \begin{bmatrix} D_p & 0 \\ 0 & 0 \end{bmatrix}, \quad M := \begin{bmatrix} M_p & 0 \\ 0 & I_{n_c} \end{bmatrix}, \quad J := \begin{bmatrix} 0 & I_{n_z} \\ 0 & 0 \end{bmatrix}, \quad G := \begin{bmatrix} D_c & C_c \\ B_c & A_c \end{bmatrix}. \tag{2.6}$$

It can be shown that the closed loop state covariance

$$X := \mathcal{E}[xx^T] \tag{2.7}$$

satisfies the following Lyapunov equation

$$(A + BGM)X + X(A + BGM)^T + (D + BGJ)W(D + BGJ)^T = 0 \tag{2.8}$$

where W is the noise covariance defined by

$$W := \mathcal{E}[ww^T] = \begin{bmatrix} W_p & W_{12} \\ W_{12}^T & V \end{bmatrix}. \tag{2.9}$$

Definition 1:

A positive definite matrix X is said to be *assignable* as a closed loop state covariance if X satisfies (2.8) for some controller G.

Since (A_p,D_p) is controllable, it can easily be shown (see for example [4]) that ($A + BGM$,$D + BGJ$) is also controllable for any choice of G provided the controller is minimal, i.e., (A_c,B_c,C_c) is controllable and observable. Therefore, by Lyapunov theory, $A + BGM$ is asymptotically stable (all its eigenvalues lie in the open left half plane of the complex plane) if and only if $X > 0$. Thus by assigning a positive definite state covariances, closed loop stability is guaranteed.

Our objective is to design a controller which yields a specified X as the closed loop state covariance. In order to accomplish this objective, we pose the following problems.

I) What are the necessary and sufficient conditions for assignability?

II) What are all controllers which assign a given assignable covariance?

III) How to construct assignable covariances?

IV) If a desirable covariance is not assignable, how to "approximate" the given desirable covariance by an element in the assignable set?

Solutions to problems I) and II) are given in the next section. Problems III) and IV) are considered in sections 4 and 5, respectively. Throughout the paper, we will assume there is no correlation between the process and the measurement noises ($W_{12} = 0$) to facilitate the presentation.

3 Necessary and Sufficient Conditions for Assignability

The following theorem presents a necessary and sufficient condition on the closed loop state covariance for assignability and gives an explicit formula for all controllers G of fixed order n_c which assign a given state covariance X.

Theorem 3.1 *There exists a G satisfying (2.8) if and only if X satisfies*

$$(I - B_pB_p^+)(A_pX_p + X_pA_p^T + D_pW_pD_p^T)(I - B_pB_p^+) = 0, \tag{3.1}$$

$$A_p\bar{X}_p + \bar{X}_pA_p^T - \bar{X}_pM_p^TV^{-1}M_p\bar{X}_p + D_pW_pD_p^T + L_pL_p^T = 0, \tag{3.2}$$

for some $L_p \in \Re^{n_p \times n_z}$ *where*

$$X := \begin{bmatrix} X_p & X_{pc} \\ X_{pc}^T & X_c \end{bmatrix}, \quad \bar{X}_p := X_p - X_{pc}X_c^{-1}X_{pc}^T. \tag{3.3}$$

If these conditions hold, all controllers G satisfying (2.8) are given by

$$\begin{aligned}
A_c &= \frac{1}{2}(X_{pc}^TM_p^TV^{-1}M_pX_{pc} - L_2L_2^T)X_c^{-1} + SX_c^{-1}, && (3.4)\\
B_c &= (L_2\hat{U}_2 - X_{pc}^TM_pV^{-\frac{1}{2}})V^{-\frac{1}{2}}, && (3.5)\\
C_c &= B_p^+(X_pM_p^TV^{-1}M_pX_{pc} - A_pX_{pc} - X_{pc}A_c^T - L_1L_2^T)X_c^{-1} + (I - B_p^+B_p)Z_1, && (3.6)\\
D_c &= B_p^+(L_1\hat{U}_2 - X_pM_p^TV^{-\frac{1}{2}})V^{-\frac{1}{2}} + (I - B_p^+B_p)Z_2, && (3.7)
\end{aligned}$$

where Z_1 *and* Z_2 *are arbitrary and*

$$\begin{aligned}
B_p &:= \begin{bmatrix} U_{B1} & U_{B2} \end{bmatrix} \begin{bmatrix} \Sigma_{B1} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{B1}^T \\ V_{B2}^T \end{bmatrix}, \quad (\,SVD\,)\\
U_x &:= U_{B2}^TX_{pc},\\
\hat{U}_1 &:= \mathsf{OF}(U_x, U_{B2}^TX_pM_p^TV^{-\frac{1}{2}}, U_{B2}^TL_p, U_1),\\
L_2 &:= X_cU_x^+U_{B2}^T(X_pM_p^TV^{-\frac{1}{2}}\hat{U}_1 - L_p) + (I - U_x^+U_x)Z_3,\\
L_1 &:= L_p + X_{pc}X_c^{-1}L_2,\\
\hat{U}_2 &:= \mathsf{OF}(B_p, L_1, X_pM_p^TV^{-\frac{1}{2}}, U_2),\\
\Theta &:= (I - B_pB_p^+)X_{pc}X_c^{-1},\\
\Phi &:= (I - B_pB_p^+)[\frac{1}{2}X_{pc}X_c^{-1}(X_{pc}^TM_p^TV^{-1}M_pX_{pc} - L_2L_2^T)\\
&\quad -X_pM_p^TV^{-1}M_pX_{pc} + A_pX_{pc} + L_1L_2^T],\\
\hat{S} &:= \Theta^+\Phi + (I - \Theta^+\Theta)[S(I - \Theta^+\Theta) - (\Theta^+\Phi)^T],
\end{aligned}$$

where Z_3 *is arbitrary and* U_1, U_2 *are arbitrary orthogonal matrices and* S *is an arbitrary skew-symmetric matrix, and the function* $\mathsf{OF}(\cdot)$ *is defined in Appendix A.*

The above results are useful for construction of assignable state covariances since (3.1) and (3.2) are uncoupled in terms of X_p and $\bar{X}_p$, and hence X_p and $\bar{X}_p$ can be obtained independently by solving the linear equation (3.1) for $X_p > 0$ and the Riccati equation (3.2) for $\bar{X}_p > 0$. However, we still have to check if $X_p \geq \bar{X}_p$ since, from (3.3), we have

$$X_p - \bar{X}_p = X_{pc}X_c^{-1}X_{pc}^T \tag{3.8}$$

and the right hand side of the above equation is nonnegative. Precise procedures to obtain an assignable covariance will be developed in the subsequent sections. Note that, from the standard monotonicity property of the stabilizing solution to the Riccati equation [5], a larger L_p will result in a larger $\bar{X}_p$.

The solution X_p to (3.1) should satisfy $X_p \geq \bar{X}_p$ for some choice of L_p in (3.2). Now, a natural question arises: What are the necessary and sufficient conditions on X_p such that there exists an L_p which yields $\bar{X}_p \leq X_p$? The following definition is needed for a more precise problem statement.

Definition 2:

A positive definite matrix X_p is said to be *assignable* as a *plant* state covariance if there exists a linear time-invariant controller which yields $\mathcal{E}[x_p x_p^T] = X_p$.

Note that $X_p > 0$ is assignable if and only if there exists an L_p which yields $\bar{X}_p \leq X_p$ and X_p satisfies (3.1). The assignability conditions for the plant state covariance is given in the following theorem.

Theorem 3.2 *A plant state covariance $X_p > 0$ is assignable if and only if X_p satisfies*

$$(I - B_p B_p^+)(A_p X_p + X_p A_p^T + D_p W_p D_p^T)(I - B_p B_p^+) = 0, \tag{3.9}$$

$$X_p \geq P, \tag{3.10}$$

where P is the positive definite solution to the following standard Riccati equation.

$$A_p P + P A_p^T - P M_p^T V^{-1} M_p P + D_p W_p D_p^T = 0. \tag{3.11}$$

Proof : Proof simply follows from the monotonicity property of the stabilizing solution to the Riccati equation.

The solution $P > 0$ to (3.11) corresponds to the estimation error covariance $P := \mathcal{E}[(x_p - x_c)(x_p - x_c)^T]$ of the Kalman filter. The assignability condition (3.10) indicates that any plant state covariance which a linear continuous-time system may possess is bounded below by a matrix that can be interpreted as the estimation error covariance of the Kalman filter. Also note that the set of all assignable plant state covariances is convex. A numerical algorithm for covariance control utilizing the convexity can be found in [1].

4 Covariance Construction for Assignability

The main result of this section is a finite step algorithm for constructing the set of all assignable state covariances via state feedback. The result of [7] shows that the set is characterized by the first assignability condition (3.9) for the dynamic output feedback case, and hence this set includes all the assignable plant state covariances via dynamic output feedback with the measurement noise, which are characterized by (3.9) and (3.10). First, we introduce a useful coordinate system for covariance construction which we call the controllable Hessenberg form.

Lemma 4.1 *Given system matrices (A,B), there exists a unitary coordinate transformation which transforms the system into controllable Hessenberg form :*

$$U^T A U = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1N} \\ B_2 & A_{22} & A_{23} & \cdots & A_{2N} \\ 0 & B_3 & A_{33} & \cdots & A_{3N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & B_N & A_{NN} \end{bmatrix}, \quad U^T B = \begin{bmatrix} B_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{4.1}$$

where

$$B_k B_k^T > 0, \quad (k = 1, 2, \cdots, N-1), \tag{4.2}$$

$$B_N B_N^T > 0, \quad or \quad B_N = 0, \tag{4.3}$$

and A_{kk} ($k = 1, 2, \cdots, N$) are square matrices.

Proof : See [6].

Note that (A,B) is controllable if and only if $B_N B_N^T > 0$, and (A,B) is stabilizable if and only if $B_N B_N^T > 0$ or A_{NN} has all its eigenvalues in the open left half plane whenever $B_N = 0$. The following theorem shows that the controllable Hessenberg form allows a recursive structure for positive definite solutions to (3.9).

Theorem 4.1 *Given* $W \geq 0$ *and system matrices* (A,B)*. Suppose* (A,B) *is in the controllable Hessenberg form given by* (4.1)*. Let* $X > 0$ *be any solution to the equation*

$$(I - BB^+)(AX + XA^T + W)(I - BB^+) = 0. \tag{4.4}$$

Then there exist matrices $\bar{X} > 0$, $R > 0$, $S = -S^T$ *and* Z *such that*

$$X = \begin{bmatrix} R + K\bar{X}K^T & K\bar{X} \\ \bar{X}K^T & \bar{X} \end{bmatrix} \tag{4.5}$$

where

$$\begin{aligned} K \; &:= \; -\frac{1}{2}\bar{B}^+(\bar{X}\bar{A}^T + \bar{A}\bar{X} + \bar{W})(2I - \bar{B}\bar{B}^+)\bar{X}^{-1} \\ &\quad + \bar{B}^+ S \bar{B}\bar{B}^+ \bar{X}^{-1} + (I - \bar{B}^+\bar{B})Z, \end{aligned} \tag{4.6}$$

and $\bar{A}$, $\bar{B}$ *are defined as*

$$A := \begin{bmatrix} A_{11} & * \\ \bar{B} & \bar{A} \end{bmatrix}, \tag{4.7}$$

and $\bar{W}$ *is the lower right corner block matrix of* W *with the same dimensions as* $\bar{A}$*. Moreover,* $\bar{X} > 0$ *satisfies*

$$(I - \bar{B}\bar{B}^+)(\bar{A}\bar{X} + \bar{X}\bar{A}^T + \bar{W})(I - \bar{B}\bar{B}^+) = 0. \tag{4.8}$$

Theorem 4.1 indicates that the assignable plant state covariance in the controllable Hessenberg coordinates has a special structure given by (4.5). The problem of solving the linear equation (4.4) can be replaced by the problem which has the same mathematical structure (see (4.8)) and smaller dimension (see (4.7)). Once we find a solution $\bar{X} > 0$ to (4.8), we can compute the solution $X > 0$ to the original problem (4.4) by choosing the free parameters: arbitrary Z, arbitrary skew-symmetric S and arbitrary positive definite R. To find solutions $\bar{X} > 0$ to (4.8), we can apply Theorem 4.1 again since $(\bar{A},\bar{B})$ is still in the controllable Hessenberg form. This yields the problem with same mathematical structure and smaller dimension again. By iteratively applying Theorem 4.1, we will eventually reach a trivial problem to solve. This idea leads to the following construction algorithm.

Theorem 4.2 *Suppose* (A_p,B_p) *is in the controllable Hessenberg form. Then the set of all assignable plant state covariance* $X_p > 0$ *via state feedback can be constructed by the following finite step algorithm. Conversely, any matrix* X_p *generated by the algorithm is assignable via state feedback. Moreover, the algorithm can generate the set of all assignable plant state covariances via dynamic output feedback as a subset in which* X_p *satisfies* $X_p \geq P$ *where* $P > 0$ *is the unique solution to (3.11).*

Construction Algorithm :

1. For $k = 1, 2, \cdots, N-1$, let

$$A(k) := \begin{bmatrix} A_{kk} & A_{kk+1} & A_{kk+2} & \cdots & A_{kN} \\ B_{k+1} & A_{k+1k+1} & A_{k+1k+2} & \cdots & A_{k+1N} \\ 0 & B_{k+2} & A_{k+2k+2} & \cdots & A_{k+2N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & B_N & A_{NN} \end{bmatrix}, \quad B(k) := \begin{bmatrix} B_k \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$W(k) := \begin{bmatrix} * & * \\ * & W(k+1) \end{bmatrix}, \quad W(k) \in \Re^{n_k \times n_k}, \quad W(k+1) \in \Re^{(n_k - r_k)\times(n_k - r_k)},$$

   where n_k and r_k are the dimensions of $A(k)$ and A_{kk}, respectively, and $A(1) = A_p$, $B(1) = B_p$, $W(1) = D_p W_p D_p^T$ and $A(N) = A_{NN}$, $B(N) = B_N$.

2. If (A_p, B_p) is controllable, choose some $X(N) > 0$ where $X(N) \in \Re^{r_N \times r_N}$. If (A_p, B_p) is stabilizable and uncontrollable, let $X(N)$ be the positive definite solution to the following Lyapunov equation.

$$A(N)X(N) + X(N)A(N)^T + W(N) = 0.$$

3. Let $k = N$.

4. Pick an arbitrary matrix $Z(k)$ and an arbitrary skew-symmetric matrix $S(k)$ and compute

$$\begin{aligned} K(k) \;&=\; -\frac{1}{2}B_p^+(k)[X(k)A^T(k) + A(k)X(k) + W(k)][2I - B(k)B^+(k)]X^{-1}(k) \\ &\quad + B^+(k)S(k)B(k)B^+(k)X^{-1}(k) + [I - B^+(k)B(k)]Z(k), \end{aligned}$$

5. Choose a positive definite matrix $R(k)$ and update X.

$$X(k-1) = \begin{bmatrix} R(k) + K(k)X(k)K^T(k) & K(k)X(k) \\ X(k)K^T(k) & X(k) \end{bmatrix}$$

6. Repeat steps 4 and 5 for $k = N, N-1, \cdots, 2$.

7. Let the resulting assignable covariance X_p be given by

$$X_p = X(1).$$

We have free parameters $Z(k)$, $S(k)$ and $R(k) > 0$ for each iteration step. $X(N) > 0$ is also free if (A_p, B_p) is controllable. The solution $X(N)$ to the Lyapunov equation in step 2 is always positive definite since $A(N)$ is asymptotically stable and $(A(N), W(N))$ is controllable.

5 Optimal Covariance Controller Design

In this section we consider the following covariance controller design problems:

1. Given a desirable plant state covariance X_d (which carries the desired system properties but may not be assignable), find an assignable $X_p > 0$ which minimizes $\|X_p - X_d\|_F$.

2. Given an assignable plant state covariance X_p, find a linear controller which assigns this X_p to the closed loop system.

Recall that the assignable plant state covariance $X_p > 0$ is characterized by two conditions (3.9) and (3.10). A tractable approach for solving problem 1 stated above is to minimize $\|X_p - X_d\|_F$ subject to (3.9) only and then to check if the resulting solution X_p satisfies (3.10). This minimization problem can be solved using the projection theorem since the first assignability condition (3.9) is a linear equation and hence the set of all homogeneous solutions to (3.9) forms a Hilbert space. Thus the solution X_p is given by the orthogonal projection of X_d onto the linear affine set spanned by all the solutions to (3.9).

Theorem 5.1 *Consider a minimization problem*

$$\min_{X_p} \|X_d - X_p\|_F \tag{5.1}$$

subject to

$$(I - B_pB_p^+)(A_pX_p + X_pA_p^T + D_pW_pD_p^T)(I - B_pB_p^+) = 0, \tag{5.2}$$

for a given X_d. The solution X_p is given by

$$X_p = H_p + \sum_{k=1}^{m} \gamma_k H_k \tag{5.3}$$

where H_p is any particular solution to (5.2) and H_k ($k = 1, 2, \cdots, m$) is the solution to

$$A_pH_k + H_kA_p^T + Y_k = 0 \tag{5.4}$$

where Y_k has 1 on the ij-element and ji-element and zero otherwise ; $j \geq i$ ($i = 1, 2, \cdots, r_B$, $j = 1, 2, \cdots, n_p$, and $r_B := rank[B_p]$) for a total of $m = r_B(n_p + \frac{1}{2}(1 - r_B))$ combinations of (i,j), and γ_k ($k = 1, 2, \cdots, m$) is given by

$$\gamma := \begin{bmatrix} \gamma_1 & \gamma_2 & \cdots & \gamma_m \end{bmatrix} = R_{xh}R_h^{-1} \tag{5.5}$$

where

$$R_{xh} := \begin{bmatrix} (\hat{X}_d, H_1) & (\hat{X}_d, H_2) & \cdots & (\hat{X}_d, H_m) \end{bmatrix} \tag{5.6}$$

$$R_h := \begin{bmatrix} (H_1, H_1) & (H_1, H_2) & \cdots & (H_1, H_m) \\ (H_2, H_1) & (H_2, H_2) & \cdots & (H_2, H_m) \\ \vdots & \vdots & \ddots & \vdots \\ (H_m, H_1) & (H_m, H_2) & \cdots & (H_m, H_m) \end{bmatrix} \tag{5.7}$$

$$\hat{X}_d := X_d - H_p \tag{5.8}$$

and (·,·) denotes the inner product on $\Re^{n_p \times n_p}$ defined by

$$(P, Q) := tr[PQ]. \tag{5.9}$$

The above theorem guarantees only one assignability condition (5.2). Therefore the other assignability condition $X_p \geq P$ ($X_p > 0$) must be checked for the dynamic output feedback case (state feedback case). The additional condition can be met by making the desired covariance X_d larger.

Now we consider the problem 2. Since we have established a method to obtain a desirable and assignable plant state covariance X_p, the next step is to find $\bar{X}_p$ satisfying

(3.2) and $\bar{X}_p \leq X_p$. Once we found $\bar{X}_p$, the controller which assigns the given state covariance can be obtained using the explicit formula given in Theorem 3.1. Therefore the crucial procedure for the solution of problem 2 is a search for $\bar{X}_p$. This can be done by choosing some $L_p \in \Re^{n_p \times n_z}$ and solving the Riccati equation (3.2) for $\bar{X}_p > 0$ and checking if the solution $\bar{X}_p$ satisfies $\bar{X}_p \leq X_p$. We summarize the procedure as follows.

Covariance Controller Design Procedure :

Given : a desirable and assignable plant state covariance X_p constructed by Theorem 5.1.

1. Pick some $L_p \in \Re^{n_p \times n_z}$ and solve (3.2) for $\bar{X}_p > 0$.
2. Check if $X_p - \bar{X}_p \geq 0$. If not, try smaller L_p. Note that $\|\bar{X}_p\|$ increases with $\|L_p\|$. Therefore, the smaller $\|L_p\|$, the more likely that $X_p \geq \bar{X}_p$ holds.
3. Find X_{pc} and $X_c > 0$ satisfying (3.8).
4. Compute the controller G which assigns this X using Theorem 3.1.

Note that choosing $L_p = 0$ yields $\bar{X}_p = P$ where P is defined in Theorem 3.2, and this $\bar{X}_p$ always satisfies $\bar{X}_p \leq X_p$ since X_p is assignable. Choosing a nonzero L_p allows additional design freedom such as the reduction of the controller order. Note that the controller order n_c is determined by

$$n_c = \text{rank}[X_p - \bar{X}_p]. \tag{5.10}$$

How to choose the free parameter L_p still remains an open issue.

6 Conclusion

New assignability conditions are derived for linear time-invariant continuous-time systems with measurement noise. Inclusion of the measurement noise yields the standard Riccati equation as the second assignability condition. Parametrizing the set of all stabilizing controllers in terms of assignable covariances, the design of linear controllers reduces to a search for an assignable state covariance which carries desirable closed loop properties. Two covariance construction problems are solved. One solution (Theorem 4.2) provides a finite step algorithm for constructing the set of all assignable plant state covariances via state feedback. The other (Theorem 5.1) finds the optimal plant state covariance which satisfies the first assignability condition (3.9) and is closest to a given desirable covariance. The freedom L_p in the second assignability condition (3.2) can be utilized to reduce the controller order. Hence, future work should determine how to choose the freedom L_p to minimize the controller order, assigning a specified plant state covariance.

Acknowledgements

The authors gratefully acknowledge many helpful discussions with A. E. Frazho and S. Swei.

A Function : Orthogonal Freedom

In this section, we define a function which is useful for stating the results of the paper.

Function : $U = \mathsf{OF}(A, L, R, F)$

Input matrices A,L and R must have the SVD's of the form.

$$(I - AA^{+})L = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{11}^T \\ V_{12}^T \end{bmatrix} \quad (SVD) \tag{A.1}$$

$$(I - AA^{+})R = [U_1 U_2] \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_{21}^T \\ V_{22}^T \end{bmatrix} \quad (SVD) \tag{A.2}$$

then the output matrix U is computed from

$$U = V_{11}V_{21}^T + V_{12}FV_{22}^T \tag{A.3}$$

In particular, if A has linearly independent rows,

$$U = F \tag{A.4}$$

References

[1] K. M. Grigoriadis and R. E. Skelton. Alternating convex projection methods for covariance control design. *Proc. Allerton Conf.*, 1992.

[2] A. Hotz and R. E. Skelton. Covariance control theory. *Int. J. Contr.*, 46(1):13–32, 1987.

[3] C. Hsieh and R. E. Skelton. All covariance controllers for linear discrete-time systems. *IEEE Trans. Automat. Contr.*, AC-35(8):908–915, 1990.

[4] T. Iwasaki and R. E. Skelton. Quadratic optimization for fixed order linear controllers via covariance control. *Proc. American Contr. Conf.*, pages 2866–2870, 1992.

[5] A. C. M. Ran and R. Vreugdenhil. Existence and comparison theorems for algebraic Riccati equations for continuous and discrete time systems. *Linear Algebra Appl.*, 99:63–83, 1988.

[6] R. E. Skelton. *Dynamic Systems Control.* Wiley, 1988.

[7] R. E. Skelton and M. Ikeda. Covariance controllers for linear continuous-time systems. *Int. J. Contr.*, 49(5):1773–1785, 1989.

[8] R. E. Skelton and T. Iwasaki. Liapunov and covariance controllers. *Proc. American Contr. Conf.*, 1992. Also to appear *Int'l J. Contr.*

[9] J. H. Xu and R. E. Skelton. An improved covariance assignement theory for discrete systems. *IEEE Trans. Automat. Contr.* to appear.

[10] K. Yasuda and R. E. Skelton. Covariance controllers: A new parametrization of the class of all stabilizing controllers. *Proc. American Contr. Conf.*, pages 824–829, 1990.

The Convergence of Asynchronous Iterations
For the Fixed Point of a Splitting Operator

Lei LI*, Tadao NAKAMURA**

* Dept. of Inform. Sys. Engin., AOMORI Univ., Aomori, 030 JAPAN.
**Dept. of Mach. Intel. & Sys. Engin., TOHOKU Univ., Sendai,980 JAPAN.

This work was supported by the Japan Science Society SASAKAWA Scientific Research Grant.

In this paper, we show the convergence of asynchronous iterations for the fixed point of the splitting operator on multiprocessors.

The problem of a nonlinear fixed point considered here is

$$X = F(X), \quad F: R^n \rightarrow R^n \tag{1}$$

It has been showed by Baudet[1], that if F is a contracting operator on a closed subset D of R^n, and if $F(D) \subset D$ then any asynchronous iteration (F,X(0),J,S) corresponding to F and starting with a vector X(0) in D converges to the unique fixed point of F in D. In this paper, we will introduce the convergence theorem of asynchronous iteration with the arbitrary splitting form for solving the nonlinear fixed point (1).

We consider the following iterative process:

$$x_i(k+1)=f_i(x_1(k_{i1}), x_2(k_{i2}), \dots, x_n(k_{in})), \quad i=1\sim n, \tag{2}$$

where $k_{ij}=k$, or $k+1$,

It is obvious that if $k_{ij}=k$, then (2) is a Jacobi iterative process, and that if

$$k_{ij} = \begin{cases} k+1 & \text{if } i>j \\ k & \text{if } i \leqq j \end{cases}$$

then (2) is a Gauss-Seidel iterative process. For these two cases, G. Baudet has discussed the convergence of asynchronous iterative scheme. But for general scheme (2), the convergence of the asynchronous iterative scheme, has not been presented yet. A mapping F^*:

$$(x_1(k), x_2(k), \dots, x_n(k))^T \rightarrow (x_1(k+1), x_2(k+1), \dots, x_n(k+1))^T$$

is decided by (2). For examples, when (2) is a Jacobi form, then F^* is F, and when (2) is a Gauss-Seidel form, then $F^*=(f_1^*,f_2^*,\dots,f_n^*)^T$ becomes

$$f_i^* = f_i(f_1^*, f_2^*, \dots, f_{i-1}^*, x_i,\dots,x_n), \quad i=1\sim n$$

Now, we always suppose that F^* exists(explicit or implicit). Suppose that F^* is the operator decided by (2):

$$X(k+1)=F^*(X(k)), \quad F^*=(f_1^*, f_2^*, \dots, f_n^*)^T : R^n \rightarrow R^n \tag{3}$$

Then (3) can be computed by the asynchronous iterative method (4) on multiprocessors

$$x_i(j) = \begin{cases} x_i(j-1), & \text{if } i \bar{\in} J_j \\ f_i^*(x_1(s_1(j)), \dots, x_n(s_n(j))), & \text{if } i \in J_j \end{cases} \tag{4}$$

We will show the following convergence theorem:

Theorem 1 (Convergence theorem of asynchronous iteration with arbitrary splitting form) If F is a contracting operator on R^n, then any asynchronous iteration (F^*, $X(0)$, J, S) with arbitrary splitting form (2) corresponding to F^* and starting with a vector $X(0)$ in R^n converges to the unique fixed point of F in R^n.

Proof: Let A be Lipchitzian matrix of F. Since F is a contracting operator, then the spectral radius $\rho(A)<1$. From the convergence theorem of asynchronous iteration presented by G.Baudet, it is obvious that Theorem 1 is correct if F^* is also a contracting operator.

For arbitrary $X(k)$, $Y(k) \in R^n$, from (3), we know

$$\| F^*(X(k))-F^*(Y(k)) \| = \| X(k+1)-Y(k+1) \|$$

And from (2) and the Lipchitzian condition, we have

$$\begin{aligned}
\| X(k+1)-Y(k+1) \| &= \| (x_1(k+1),\dots,x_n(k+1))^T-(y_1(k+1),\dots,y_n(k+1))^T \| \\
&= \| (f_1(x_1(k_{11}),\dots,x_n(k_{1n})),\dots, f_n(x_1(k_{n1}),\dots,x_n(k_{nn})))^T \\
&\quad -(f_1(y_1(k_{11}),\dots, y_n(k_{1n})), \dots, f_n(y_1(k_{n1}),\dots,y_n(k_{nn})))^T \| \\
&\leqq \Big(\sum_{j=1}^{n} a_{ij} \,|\, x_j(k_{1j})-y_j(k_{1j}) \,|\, ,\dots, \sum_{j=1}^{n} a_{nj} \,|\, x_j(k_{nj})-y_j(k_{nj}) \,|\, \Big)^T \\
&= \Big(\sum_{j=1}^{n} a_{1j}(1) \,|\, x_j(k+1)-y_j(k+1) \,|\, ,\dots, \sum_{j=1}^{n} a_{nj}(1) \,|\, x_j(k+1)-y_j(k+1) \,|\, \Big)^T \\
&\quad + \Big(\sum_{j=1}^{n} a_{1j}(2) \,|\, x_j(k)-y_j(k) \,|\, ,\dots, \sum_{j=1}^{n} a_{nj}(2) \,|\, x_j(k)-y_j(k) \,|\, \Big)^T \\
&= A(1) \| X(k+1)-Y(k+1) \| + A(2) \| X(k)-Y(k) \|
\end{aligned}$$

where

$$a_{ij}(1) = \begin{cases} a_{ij}, & \text{if } k_{ij}=k+1 \\ 0, & \text{if } k_{ij}=k \end{cases}$$

$$a_{ij}(2) = \begin{cases} 0, & \text{if } k_{ij}=k+1 \\ a_{ij}, & \text{if } k_{ij}=k \end{cases}$$

$$A(1)= (a_{ij}(1))_{n*n}, \qquad A(2)= (a_{ij}(2))_{n*n}$$

therefore $(I-A(1)) \| X(k+1)-Y(k+1) \| \leqq A(2) \| X(k)-Y(k) \|$

Because $\rho(A)<1$ and A is a nonnegative matrix, $I-A$ is a M-matrix [4]. From the properties of M-matrix, we know that $I-A_1$ is also a M-matrix, and that $(I-A_1)^{-1}$ is a nonnegative matrix. So we have

$$\| X(k+1)-Y(k+1) \| \leqq (I-A(1))^{-1}A(2) \| X(k)-Y(k) \|$$

$$\| F^*(X(k))-F^*(Y(k)) \| = \| X(k+1)-Y(k+1) \| \leqq (I-A(1))^{-1}A(2) \| X(k)-Y(k) \|$$

but from the properties of M-matrix, $\rho((I-A(1))^{-1}A(2))<1$, therefore, F^* is a contracting operator.

References

[1] Baudet, G.M., Asynchronous iterative methods for multiprocessors, J Assoc. Comput. Mach., 25(2): 226-244(1978).

[2] Tarazi, M.N., Some convergence results for asynchronous algorithms, Numer. Math., 39: 325-340(1982).

[3] Li Lei, Convergence of asynchronous iteration with arbitrary splitting form, Linear Algebra Appl., 113: 119-127(1989).

[4] Henryk,M., Nonnegative Matrices, John Wiley & Sons, 1988.

MULTI-TARGET NARROWBAND DIRECTION FINDING AND TRACKING USING MOTION DYNAMICS

Anuj Srivastava and Nicholas Cutaia and
Michael I. Miller and Joseph A. O'Sullivan and Donald L. Snyder

1 Introduction

Multiple target tracking and narrowband direction finding are well known problems in the signal processing literature. In [1] we have described a new family of random sampling algorithms for use in multi-target tracking of dynamically moving targets by passive sensor arrays. We take a Bayesian approach where we define a posterior probability over the configuration space of multiple tracks and use a jump-diffusion algorithm to generate conditional mean estimates from the probability measure. The mathematical foundation for jump-diffusion processes used for model-order estimation problems of this kind is developed in [2, 3].

In this paper we present a new prior on track formation which is based on the dynamics of target motion. The force equations governing the motion of targets are utilized to form a prior density on the multi-track parameter space. This combined with the likelihood provides the posterior measure for the jump-diffusion algorithm.

Section 2 analyzes the aircraft motion so as to induce the prior on track formation, with section 3 deriving the signal model and the Bayes posterior.

All authors are associated with Electronic Signals and Systems Research Laboratory of the Department of Electrical Engineering at Washington University, St. Louis , Missouri 63130. This work was supported by Rome Laboratory contract number F30602-92-C-0004, by the ARO P-29349-MA-SDI and ONR 59922.

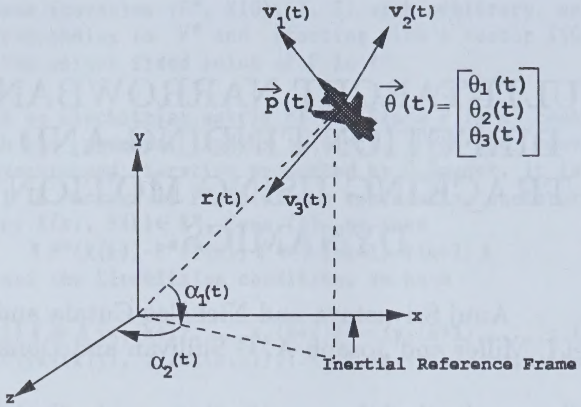


Figure 1: Aircraft motion in 3 space, $\vec{\theta}(t)$ representing the rotational motion while $\vec{p}(t)$ signifies the translational motion.

Section 4 shows various results on multiple target tracking using the jump-diffusion algorithm.

2 Motion Dynamics Induced Prior

As an object moves in 3 space it traverses a continuous path consisting of a sequence of point locations $\vec{p}(t) = [p_1(t), p_2(t), p_3(t)]^T \in \Re^3$ as shown in Figure (1). For describing their dynamics we shall be interested in expressing the tracks in terms of the velocities of the objects, $\vec{v}(t) = [v_1(t), v_2(t), v_3(t)]^T \in \Re^3$ which are related to the positions according to $\vec{p}(t) = \int_{t_0}^{t} \vec{v}(\tau)d\tau + \vec{p}(t_0)$.

In defining the prior we make the following assumptions: (i) the target is a rigid body, (ii) the earth's curvature is negligible, (iii) the earth's motion is negligible compared to target motion, and (iv) wind effects are negligible. Under these conditions, the translational motion of the target is governed by the force equations based on Newton's second law of motion, given by

$$\dot{v}_1(t) + q_2(t)v_3(t) - q_3(t)v_2(t) \;=\; \frac{1}{m}(f_1(t) - wsin(\theta_1(t))),$$

$$\begin{aligned} \dot{v}_2(t) + q_3(t)v_1(t) - q_1(t)v_3(t) &= \frac{1}{m}(f_2(t) + w sin(\theta_2(t))cos(\theta_1(t))), \\ \dot{v}_3(t) + q_1(t)v_2(t) - q_2(t)v_1(t) &= \frac{1}{m}(f_3(t) + w cos(\theta_2(t))cos(\theta_1(t))) \,. \end{aligned}$$

Each equation corresponds to the motion along one of the principle axes. In these equations, $f_1(t), f_2(t)$, and $f_3(t)$ are the applied linear forces on the target, w and m are the weight and mass of the target, $\vec{\theta}(t) = (\theta_1(t), \theta_2(t), \theta_3(t)) \in [0, 2\pi)^3$ are the Euler angles representing the orientation of the target with respect to its body frame of reference, and $\vec{q}(t) = [q_1(t), q_2(t), q_3(t)]$ are the rates of change of orientation, which are functions of Euler's angles and their derivatives. In vector form this becomes

$$\dot{\vec{v}}(t) + A(\vec{\theta}(t))\vec{v}(t) = \vec{f}(t), \tag{1}$$

where

$$A(\vec{\theta}(t)) = \begin{bmatrix} 0 & -q_3(t) & q_2(t) \\ q_3(t) & 0 & -q_1(t) \\ -q_2(t) & q_1(t) & 0 \end{bmatrix},$$

and

$$\vec{f}(t) = \frac{1}{m} \begin{bmatrix} f_1(t) - w sin(\theta_1(t)) \\ f_2(t) + w sin(\theta_2(t))cos(\theta_1(t)) \\ f_3(t) + w cos(\theta_2(t))cos(\theta_1(t)) \end{bmatrix}.$$

This linear differential equation is characterized by the time-varying parameter matrix $A(\vec{\theta}(t))$ and force vector $\vec{f}(t)$. We utilize the vector equation to determine the covariance of the process $\vec{v}(t)$. To do this, we follow the approach of Amit et al. [4] by assuming the forcing function is a white process with mean $\bar{\vec{f}}(t)$ and a fixed spectral density σ, which then induces a Gaussian process $\vec{v}(t)$ with mean $\bar{\vec{v}}(t)$ and the covariance operator determined by the differential operator of Eqn. (1). The mean and covariance of the velocity process are given by

$$\begin{aligned} \bar{\vec{v}}(t) &= \int_{t_0}^{t} e^{-\int_{t_1}^{t} A(\tau)d\tau} \bar{\vec{f}}(t_1)dt_1 + \bar{\vec{v}}(t_0), & (2) \\ K_v(t,s) &= \sigma \int_{t_0}^{min(t,s)} [e^{-\int_{t_1}^{t} A(\tau)d\tau}][e^{-\int_{t_1}^{s} A(\tau)d\tau}]^T dt_1. & (3) \end{aligned}$$

The covariance is parameterized by the sequence of airplane orientations $\{\vec{\theta}(\tau)\}_{t_0}^{min(t,s)}$ which is modeled as a continuous Markov process. For all of the implementation we use the Markov Von-Mises prior used in [1] but extended to the sphere.

3 The Posterior

As we assume that the sensor array for narrowband tracking provides a discrete set of sample measurements, we discretize the path and the associated velocities as follows. Defining the point in three-space at which the target resides at the l-th time sample $l\Delta$ as $\vec{p}[l]$, then the velocities are assumed to be piecewise constant given by their values at the sample points $l\Delta, l = 1, 2, \ldots, L$ so that $\vec{p}(l) = \sum_{l=1}^{l} \vec{v}(l)\Delta + \vec{p}(0)$.

Now the discrete prior and likelihood are defined.

3.1 Prior

Using the mean's and covariance's model from Eqns. (2,3) the prior on the discrete set of velocity vectors is derived straightforwardly. First we discretize the equation of motion according to

$$\vec{v}[n+1] = (I - A[n])\vec{v}[n] + \vec{f}[n], \tag{4}$$

where $A[n]$ implies $A(\vec{\theta}[n])$. Now define the covariance of the L-length velocity vector of the m-th track $V^{(m)} = \left(\vec{v}^{(m)}[1]\ , \vec{v}^{(m)}[2]\ , \ldots\ , \vec{v}^{(m)}[L]\right)^T \in \Re^{3L}$ as matrix $K_{V^{(m)}} = E\{(V^{(m)} - \bar{V}^{(m)})(V^{(m)} - \bar{V}^{(m)})^T\}$, which is an $L \times L$ block matrix with 3×3 sub-block elements. As the driving function is assumed white with covariance $E\{(\vec{f}[j] - \bar{\vec{f}}[j])(\vec{f}[k] - \bar{\vec{f}}[k])^T\} = \sigma I \delta[j-k]$, the element $(K_{V^{(m)}})_{j,k}$ with block indices j,k is given by

$$E\{(\vec{v}[j] - \bar{\vec{v}}[j])(\vec{v}[k] - \bar{\vec{v}}[k])^T\} = \sigma \sum_{i=1}^{min(j,k)-1} M[i+1, j-1]M^T[i+1, k-1]\,. \tag{5}$$

where $M[j+1, n]$ is the 3×3 matrix

$$M[j+1, n] = \prod_{l=j+1}^{n} (I - A[(n+j+1) - l]),$$

and $M[n+1,n]$ is defined to be the identity matrix. The mean velocity vector at any time instant $n+1$ becomes

$$\vec{\bar{v}}[n+1] = \sum_{j=1}^{n} M[j+1,n]\vec{\bar{f}}[j] + M[1,n]\vec{\bar{v}}[1].$$

Given the mean velocities and the covariance matrix, the prior on the m-th track $(V^{(m)}, \Theta^{(m)})$ can be written as

$$\pi(V^{(m)}, \Theta^{(m)}) = (2\pi)^{\frac{-3L}{2}} det^{\frac{-1}{2}} [K_{V^{(m)}}]\, e^{-\frac{1}{2}\{(V^{(m)}-\bar{V}^{(m)})^T K^{-1}_{V^{(m)}} (V^{(m)}-\bar{V}^{(m)}) + E(\Theta^{(m)}\}}, \tag{6}$$

where $E(\Theta^{(m)})$ represents the prior energy on the rotational motion of the m-th target described by its set of Euler's anlges.

3.2 Signal Model & Data Likelihood

To describe the multi-track, multi- sensor scenario we use the standard narrowband signal model developed in [5] and used for tracking in [1], [6]. The targets are assumed observed using a narrowband sensor array, in the experiments shown here a uniform cross array consisting of two uniform linear arrays orthogonal to each other lying in the inertial frame of reference Figure (1). This array is sensitive to the range, elevation, and azimuth locations of the targets $\{\vec{\alpha}(t) : r(t), \alpha_1(t), \alpha_2(t)\}$, respectively. These angles are measured in the inertial frame of reference while the previously defined velocities on which the prior is based are described in the body centered frame. The variables in the two reference frames are related by known transformations [7]. Notice, using the standard cartesian to polar coordinate transformation and the above mentioned transformations, the azimuth, elevation, and range parameters are deterministically related to the velocities. Define a transformation Φ from velocities in body centered frame to the point locations in the inertial frame as

$$\Phi : \{\vec{v}(l)\}_{l=1}^{L} \in \Re^{3L} \longmapsto \{\vec{\alpha}(\tau)\}_{l=1}^{L} \in [0, 2\pi)^{2L} \times \Re_{+}^{L}$$

The transformation Φ allows us to write the likelihood of the sensor data in terms of the velocity vectors.

For the m-th source the data collected at the P-element sensor array at discrete time l is the superposition of the incoming signal and the ambient noise, and is given by

$$\mathbf{y}_m[l] = \mathbf{d}(\Phi(\vec{v}^{(m)}[l]))s_m[l] + \mathbf{n}_m[l], \tag{7}$$

where $\mathbf{n}$[l] is a $P \times 1$, 0-mean complex gaussian random vector with identity covariance, $s_m[k]$ is the signal value and $\mathbf{d}(\Phi(\vec{v}^{(m)}[l]))$ is a regular $P \times 1$ Vandermonde direction vector with the angles of arrival parameterized by the velocity vector $\vec{v}^{(m)}[l]$. In the multiple target case, the received signal is a superposition of the signals from each target. Assuming there are M targets present, the received data in matrix form becomes

$$\mathbf{Y}[l] = \mathbf{D}_M(\Phi(\vec{v}^{(1)}(l)),..,\Phi(\vec{v}^{(M)}(l)))\mathbf{s}_M[l] + \mathbf{n}[l] \tag{8}$$

where $\mathbf{D}_M$ is the $P \times M$ matrix $[\mathbf{d}(\Phi(\vec{v}^{(1)}[l]))\ \mathbf{d}(\Phi(\vec{v}^{(2)}[l]))\ ...\ \mathbf{d}(\Phi(\vec{v}^{(M)}[l]))]$, and $\mathbf{s}_M[l]$ is the $M \times 1$ vector of signal amplitudes at time l.

The vector of signal values $\mathbf{s}_M[l]$ is modeled as a deterministic quantity, with the likelihood for an M-track scene becoming

$$L(data|V^{(1)},\ldots,V^{(M)}) \propto \prod_{l=1}^{L} e^{-|\mathbf{Y}[l]-\mathbf{D}_M(\Phi(\vec{v}^{(1)}(l)),..,\Phi(\vec{v}^{(M)}(l)))s_M[l]|^2}. \tag{9}$$

3.3 Parameter Space and Posterior

To formulate the Bayes posterior first we define the parameter space on which it has support, and then combine the prior from track dynamics with the likelihood.

Define a track-configuration as the set of parameters which completely describe the scene at any time. Suppose in an observation period the motion of the *m-th* source is sampled L times so its track is completely defined by a sequence of L velocity vectors according to, $V^{(m)} = (\vec{v}^{(m)}[1],\ldots,\vec{v}^{(m)}[L])^T \in \Re^{3L}$. For a multiple track scene with M targets, M being unknown, the configuration is the set of M tracks $(V^{(1)},\ldots,V^{(M)}) \in \Re^{3ML}$, with the complete configuration or parameter space being $\bigcup_{M=0}^{\infty} \Re^{3ML}$. The parameter space is a countable, infinite union of subspaces each having an associated value of M, highlighting an important issue in this estimation problem. The model order, i.e. the number M of targets present in the observation space, is not known beforehand. Each scene of M targets has a variable number 3ML of parameters, with M itself a parameter to be estimated by the algorithm.

Since the prior on motion dynamics involves the trajectories of the Euler angles $\vec{\theta}$, a bigger parameter space is required to account for both the rotational and translational motions . The parameter set for M-tracks becomes $((V^{(1)}, \Theta^{(1)}), \ldots, (V^{(M)}, \Theta^{(M)})) \in \Re^{3ML} \times [0, 2\pi)^{3ML}$, with the complete parameter space being $\bigcup_{M=0}^{\infty} \Re^{3ML} \times [0, 2\pi)^{3ML}$. Now we define the posterior measure over this complete configuration space.

For the multi-target case, the target motion is assumed to be independent across targets, giving the prior

$$\pi((V^{(1)}, \Theta^{(1)}), (V^{(2)}, \Theta^{(2)}), ..., (V^{(M)}, \Theta^{(M)}, M) = \prod_{m=1}^{M} \pi(V^{(m)}, \Theta^{(m)})\ \pi(M)\ , \tag{10}$$

with $\pi(M)$ the prior on the number of tracks and $\sum_{M=0}^{\infty} \pi(M) = 1$.

Combining the likelihood and prior terms the posterior is of Gibbs form $\pi = \frac{e^{-U}}{Z}$ with potential

$$\begin{aligned} U(V^{(1)}, \Theta^{(1)}), (V^{(2)}, \Theta^{(2)}), \ \ldots\ &, (V^{(M)}, \Theta^{(M)}, M) \\ &= \sum_{l=1}^{L} (-|\mathbf{Y}[l] - \mathbf{D}_M(\Phi(\vec{v}^{(1)}[l], ..\Phi(\vec{v}^{(M)}[l])))s_M[l]|^2) \\ &- \sum_{m=1}^{M} [(V^{(m)} - \bar{V}^{(m)})^T K_{V^{(m)}}^{-1} (V^{(m)} - \bar{V}^{(m)})] + \sum_{m=1}^{M} E(\Theta^{(m)}) \\ &+ \ log(\pi(M)). \end{aligned}$$

4 Simulation Results

The estimation is performed by sampling the multi-track parameter space $\bigcup_{M=0}^{\infty} \Re^{3ML} \times [0, 2\pi)^{3ML}$ using jump-diffusion processes as described in [1, 2, 3]. We construct a Markov process which visits the candidate solutions according to the posterior Gibb's density $\pi = \frac{e^{-U}}{Z}$. In this algorithm, the jump process carries the estimate from subspace to subspace of various track numbers, with the diffusion following stochastic gradient descent within the subspaces. It has been proven that the jump-diffusion produces samples from the posterior, so that the empirical averages converge to their conditional mean expectations under the posterior.

Figure (2) demonstrates the use of jump-diffusion based sampling algorithm for multiple target tracking in the plane where the tracks are com-

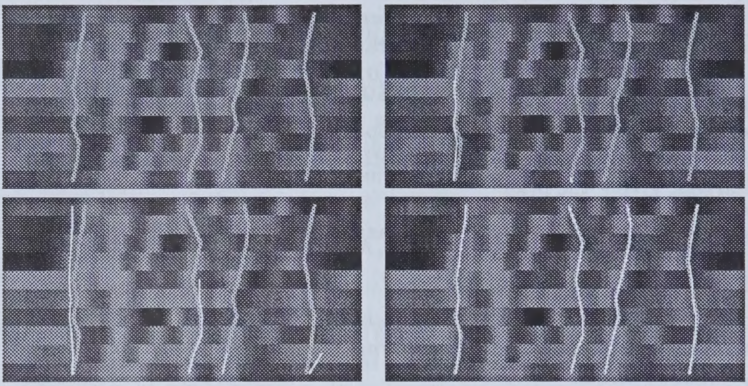


Figure 2: Illustrating jump-diffusion based sampling for 2D problem estimating only one angular location. In the background is the spatial power spectrum with brightness representing energy at that angular location. The top left panels shows the four tracks, top right and bottom left show intermediate stages of estimation with the final result in the bottom right.

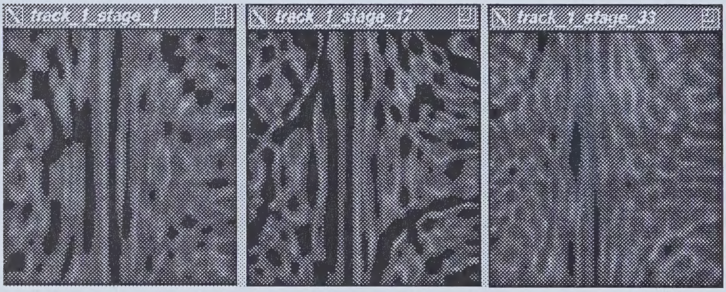


Figure 3: The upper panel displays the spatial power spectra at three locations along the track in 3 space with brightness a function of energy.

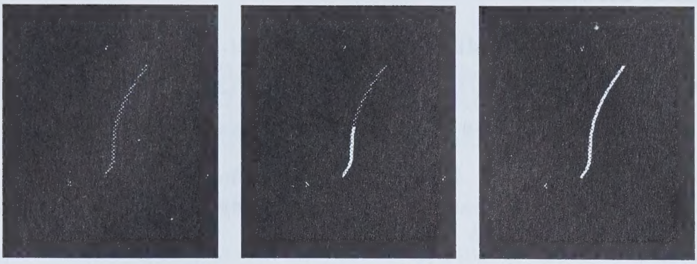


Figure 4: Estimation of single track configuration in 3D space. The left panel shows the true track in gray. The middle and right panels show the intermediate and final track estimates with the estimated tracks overlapping in white.

pletely specified via a single elevation angle. The prior used is the Von-Mises prior (see [1] for details of the implementation etc). Further,the targets are observed by a uniform linear array, with the algorithm forced to estimate the number of targets as well as the paths themselves. The four panels show various stages of the algorithm. The actual tracks are shown in top left while top right and bottom panels show the intermediate and the final results with the estimates drawn over the actual tracks. Shown in the background are the spatial power spectra of the data generated using *minimum variance distortionless response* (MVDR) beamforming [8] with the brighter spots signifying higher energy locations.

For 3D problem, the algorithm was run on a simulated data set with track length L = 64 and a cross-array having two 32 elements subarrays each with half wavelength spacing. Shown in Figure (3) are a series of spatial power spectra of data at three different time indices along the track. Figure (4) shows the estimation of a complete track from this noisy data set. The left panel shows the actual track in gray in the inertial frame of reference with the middle and right panels showing the estimated track drawn over the actual track at the successive stages of the algorithm.

References

[1] A. Srivastava, M. I. Miller, and U. Grenander. Jump-diffusion processes for object tracking and direction finding. In *Proc. 29th Annual Allerton Conference on Communication ,Control and Computing*, pages 563–570, Urbana, Il, October 1991. University of Illinois.

[2] U. Grenander and M. I. Miller. Jump-diffusion processes for abduction and recognition of biological shapes. *Monograph of the Electronic Signals and Systems Research Laboratory*, 1991.

[3] U. Grenander and M. I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society*, in review.

[4] Y. Amit, U. Grenander, and M. Piccioni. Structural image restoration through deformable templates. *J. American Statistical Association*, 1991.

[5] R. Schmidt. *A signal subspace approach to multiple emitter location and spectral estimation.* Ph.D. Dissertation of Stanford University, Palo Alto, CA., Nov. 1981.

[6] M.I. Miller and D. R. Fuhrmann. Maximum likelihood narrow-band direction finding and the em algorithm. *IEEE Acoust. Speech and Signal Processing*, 38, No.9(38, No.9):560–577, 1990.

[7] Bernard Friedland. *Control System Design : An Introduction To State-Space Methods.* McGraw-Hill Book Company, 1986.

[8] N. Owsley. Sonar array processing. In Simon Haykin, editor, *Array Signal Processing.* Prentice-Hall, New Jersey, 1985.

Optimum Quantization in Single and Multisensor Random Signal Detection *

Rick S. Blum
Department of Electrical Engineering and Computer Science
Lehigh University
Bethlehem, PA 18015

Abstract

Optimum single and multisensor detection schemes based on quantized data are of great interest in radar and sonar applications. The design and properties of such schemes are considered here for cases with weak random signals, asymptotically large sample sizes, and unknown signal-to-noise ratios. Our results indicate that the best schemes for symmetric problems sometimes use quantizer characteristics that are not symmetric. For the case of independent signal observations taken at multiple sensors, we demonstrate how the optimum schemes can be found from the optimum schemes for single sensor cases. We also demonstrate that when a quantizer is located at each sensor it is sometimes best for each quantizer to use different size alphabets. In the case of a common signal at each sensor, the optimum schemes are shown to be different from those for the independent signal and the single sensor cases.

I. Introduction

In many situations it is desirable or necessary to detect signals based on quantized data [1]-[5] and thus to use quantized detector characteristics. Procedures are needed to design these quantized detector characteristics so that detection schemes which use them can provide optimum performance. Such procedures have been addressed in the past but usually under the assumption of symmetric quantizer characteristics. Many researchers have assumed that optimum quantized detection schemes for symmetric problems would always use symmetric quantization characteristics. While frequently this is a valid assumption our results indicate that performance can sometimes be improved through consideration of nonsymmetric quantizers.

The nonsymmetry of optimum detection quantizers has been demonstrated for some known signal detection problems in the past [3]. We investigate this phenomenon for a symmetric single sensor random signal detection problem with an even number of levels.

---

*This material is based upon work supported by the National Science Foundation under Grant No. MIP-9211298

We consider the canonical case of detecting a random signal in possibly nonGaussian noise with a set of statistically independent observations for which the signal-to-noise ratio is unknown. So that closed form solutions may be found easily, we focus on the case of large observation sample sizes and weak signals, called asymptotically optimum (AO) quantization.

Since quantization is useful in multisensor detection schemes we also consider AO quantization schemes for detecting a random signal in possibly nonGaussian noise using multiple sensors. Very few results giving optimum quantization schemes exist for this case. Some results on AO quantization for a multisensor random signal detection problem are given in [4] but the form of the tests used there are generally suboptimum. While the schemes given in [4] often lead to less complicated design procedures than those given here, our schemes generally yield better performance under the same constraints.

Section II reviews the necessary conditions for an AO quantization scheme used to detect a random signal in additive noise with observations from a single sensor. We find solutions for some specific cases with generalized Gaussian noise probability density functions (p.d.f.s) and for quantizers which use different number of levels. The number of levels in a quantizer indicates its alphabet size. We demonstrate that for quantizers with an even number of levels the best solutions are usually nonsymmetric, while the best solutions with an odd number of levels are usually symmetric. We also show that the necessary conditions given in Section II do not have an even symmetric solution for many cases with an even number of levels. We show that our asymptotically best nonsymmetric quantizers perform better than the asymptotically best symmetric quantizers in some specific cases with a finite observation sample size and non-zero signal strengths.

Section III studies cases with multiple sensors where a quantizer is located at each sensor and the quantized observations are used to detect a random signal in additive noise. In Part A we show that if the signal observations are independent then AO quantization schemes can be found using the results for the single sensor cases discussed in Section II. We also demonstrate that if the sensor quantizers can use a fixed number of total levels in any manner they choose, it is sometimes better to use quantizers with a different number of levels at each sensor. In Part B of Section III we describe an AO quantization scheme for the case of a common random signal at each sensor. An interesting aspect of these results is that the optimum quantizers and the method of fusing the quantized observations are generally different from both the single sensor results and the results for independent signal observations.

II. Single Sensor Results

Let $\underline{X} = (X_1, X_2, \ldots, X_n)$ be a vector of independent real-valued random variables. Based on $\underline{X}$, we wish to decide between

$$H_0 : X_i = N_i; \qquad i = 1, \ldots, n \quad \text{(signal absent)} \tag{1}$$

vs.

$$H_1 : X_i = \theta S_i + N_i; \qquad i = 1, \ldots, n \quad \text{(signal present } \theta \neq 0) \tag{2}$$

where $\{N_i; i = 1, \ldots, n\}$ is an independent, identically distributed (i.i.d.) zero-mean noise sequence with a known common univariate p.d.f. f, θ is a real scalar parameter which represents the signal strength, and $\{S_i; i = 1, \ldots, n\}$ is a zero-mean, independent random signal sequence which is independent from the noise samples. With some regularity conditions on the p.d.f. f [4], a test using a test statistic of the form

$$\tau(\underline{x}) = \sum_{i=1}^{n} E[S_i^2] g_o(x_i), \tag{3}$$

with $g_o(x) = \frac{f''(x)}{f(x)}$ yields best performance for weak signals and is called locally optimum (LO) with $g_o(x)$ called the LO nonlinearity. A LO test is also AO [4] and an AO test is one which provides maximum detection efficacy [1]. The efficacy of a test is given by

$$\xi = \lim_{n \to \infty} \frac{[\frac{\partial^2}{\partial^2 \theta} E_\theta\{T_n(\underline{x})\}|_{\theta=0}]^2}{n Var_{\theta=0}(T_n(\underline{x}))} \tag{4}$$

when one uses a test statistic T_n which is based on a set of n observations $\underline{x}$ each described by (1) or (2) to decide between H_1 and H_0. Now consider test statistics based on quantized data as described by the test statistic

$$\tau_Q(\underline{x}) = \sum_{i=1}^{n} E[S_i^2] Q(x_i) \tag{5}$$

where Q is an M level quantizer. The quantizer levels q_k and breakpoints t_k, with $t_0 = -\infty$ and $t_M = \infty$, are defined as

$$Q(x) = q_k \qquad \text{when } x \in (t_{k-1}, t_k]; \qquad k = 1, \ldots, M \tag{6}$$

The necessary conditions for the asymptotic optimality of these quantizers are [1]

$$q_k = \frac{\int_{t_{k-1}}^{t_k} g_o(x) f(x) dx}{\int_{t_{k-1}}^{t_k} f(x) dx}; \qquad k = 1, \ldots, M \tag{7}$$

and

$$g_o(t_k) = \frac{(q_k + q_{k+1})}{2}; \qquad k = 1, \ldots, M-1 \tag{8}$$

Equations (7) and (8) are also necessary conditions to minimize the mean-square error between the quantizer and the LO nonlinearity and it is demonstrated in [2] that an AO quantizer will provide a minimum mean-square-error "fit" to the LO nonlinearity.

New Results

We investigated the optimum detection quantizer designs, without assuming symmetric quantizer characteristics, for detecting a weak random signal in zero-mean, unit-variance generalized Gaussian noise with the p.d.f.

$$f(x) = \frac{k}{2\mathcal{A}(k)\Gamma(1/k)} \exp -[|x|/\mathcal{A}(k)]^k \tag{9}$$

where

$$\mathcal{A}(k) = \left(\frac{\Gamma(1/k)}{\Gamma(3/k)}\right)^{\frac{1}{2}} \tag{10}$$

and where $\Gamma(a) = \int_0^\infty x^{a-1}e^{-x}dx$ is the gamma function. We attempted to find the AO detection quantizers for cases with generalized Gaussian densities with $k = 1.6, 1.8,$ and 2.0 and for cases where the number of levels in the quantizer M vary from 3 to 8. Our procedure was to search for all solutions to the necessary conditions and then to compare the detection efficacy of each solution to find the best. Figure 1 shows the best 5-level quantizer characteristic we found and the LO nonlinearity for a generalized Gaussian noise p.d.f. with $k = 2$ (the Gaussian case). The efficacy of this quantizer, a measure of weak-signal and large observation sample size detection capability, was 1.6488 and in this case our best quantizer was symmetric. Figure 1 also shows the best 6-level quantizer characteristic we found for a Gaussian noise p.d.f. and in this case our best quantizer was nonsymmetric. The efficacy of our best 6-level quantizer was 1.7175 so that the addition of the extra quantization level, which made the optimum quantizer characteristic nonsymmetric, improved performance. Both quantizers attempt to provide a minimum mean-square-error fit to the LO nonlinearity but the nonsymmetric quantizer with 6 levels provides a slightly better fit and therefore slightly better performance. Figure 1 clearly illustrates that this better fit is accomplished by obtaining a closer match to one side of the LO nonlinearity by allocating the "extra" level to that side. Figure 2 shows our best 3-level and 4-level quantizer characteristics and the LO nonlinearity for a generalized Gaussian noise p.d.f. with $k = 1.6$. In this case the 3-level quantizer provides an efficacy of 2.017 while the 4-level quantizer has an efficacy of 2.102. We have obtained similar results for different size quantizers and for different generalized Gaussian p.d.f.s. For cases with an odd number of levels, we found that a symmetric and possibly several nonsymmetric solutions to the necessary conditions will exist, with the symmetric solution yielding best performance. For cases with an even number of quantization levels, we found only nonsymmetric solutions to the necessary conditions. We have demonstrated analytically that an even symmetric quantizer solution to these necessary conditions does not exist in many cases which have an even number of quantization levels. These cases include all those where the LO nonlinearity is even symmetric with a global minimum at zero, including cases which do not have generalized Gaussian additive noise and which may not be random signal detection problems. The proof of this result is given in Appendix A.

While the apparent nonsymmetry of the optimum quantizers for a random signal detection problem may seem odd at first, it is actually quite easy to understand. To obtain an even symmetric quantizer with an even number of levels, a quantizer which really has one less level must be used. It is not surprising that the unused level can be used to achieve better performance with a quantizer which is nonsymmetric and this appears to be the case in the examples given in this section. It is certainly reasonable that in some cases a nonsymmetric quantizer can provide a better fit to the LO nonlinearity by using the extra level.

While nonsymmetric quantizers have been shown to be superior in some cases we have seen that the performance improvement over the best symmetric quantizer is usually quite small. One may wonder if this improvement is an artifact of focusing on asymptotic performance. We have also studied the performance of our quantizers at non-zero signal strengths via Monte Carlo simulations with 1 million runs. We simulated the case of a random signal in (2) that has either the value $S_i = +1$ or the value $S_i = -1$ with equal probabilities and we considered only a sample size of $n = 100$. We compared the performance of our best symmetric and nonsymmetric quantizers with 4, 6, and 8 levels for Gaussian noise. In all cases we found that our best nonsymmetric quantizers yielded better performance. The Monte Carlo simulation results for the 4-level quantizer case, in the form of probability of detection as a function of signal strength θ, appear in Figure 3. The difference in performance decreased for cases with a larger number of levels. From these results we recognize that the difference in performance between the best symmetric and nonsymmetric quantizers is frequently quite small, especially for a reasonably large number of quantization levels. For cases where the extra performance is required, use of a nonsymmetric quantizer may be justified. In many cases, however, the lower complexity of a symmetric quantizer will make it more desirable.

III. Multiple Sensor Results

Consider a case with N sensors where the observations at each sensor are each individually quantized and then sent to a fusion center. After n quantized observations have been sent from each sensor to the fusion center, the fusion center makes a final signal detection decision based on all the data received. Denote the i^{th} discrete-time observation at the p^{th} sensor by $X_i^p; i = 1, \ldots, n \; p = 1, \ldots, N$. This observation will be quantized by the p^{th} sensor quantizer Q^p which will produce one of M_p possible symbols $1, \ldots, M_p$ (each symbol corresponds to a level in the single sensor case). Q^p is completely described by the $M_p + 1$ breakpoints $t_j^p; j = 0, \ldots, M_p$ with $t_0^p = -\infty$ and $t_M^p = \infty$. Q^p produces the j^{th} symbol at discrete-time i if X_i^p is in the j^{th} quantization interval so that $t_{j-1}^p < X_i^p \leq t_j^p$ and this operation can be summarized by the $n \times M_p$ random matrix $\mathbf{Z_p}$. Each row of this matrix corresponds to a different time sample and each column corresponds to a different quantization symbol. If the p^{th} sensor quantizer produces the j^{th} symbol at discrete-time i, then the (i, j) entry of the matrix $\mathbf{Z_p}$, which we denote by z_{ij}^p, will be 1 and all other entries of the i^{th} row of $\mathbf{Z_p}$ will be 0.

We consider a random signal detection problem where the observation at the i^{th} discrete-time instant at the p^{th} sensor is given by

$$X_i^p = \theta S_i^p + N_i^p; \qquad i = 1, \ldots, n \tag{11}$$

where $\{N_i^p; i = 1, \ldots, n \; p = 1, \ldots, N\}$ is a set of i.i.d. zero-mean noise samples with a known common univariate p.d.f. f, θ is a parameter which represents the signal strength, and $\{S_i^p; i = 1, \ldots, n \; p = 1, \ldots, N\}$ are zero-mean, random signal samples which are independent from the noise samples. We shall investigate the case where the S_i^p are all i.i.d. and the case of a common signal at each sensor $S_i^p = S_i$; $p = 1, \ldots, N$ where the S_i are i.i.d. The testing problem is again to determine whether signal is present $H_1 : \theta \neq 0$ or absent $H_0 : \theta = 0$.

Since $\mathbf{Z_p}$ represents the quantized observations from the p^{th} sensor, the LO fusion test statistic λ_{LO} based on the quantized observations can be computed as [4]

$$\lambda_{LO} = \frac{\frac{\partial^2}{\partial \theta^2} P_{\mathbf{Z_1}, \ldots, \mathbf{Z_N}}(\mathbf{z_1}, \ldots, \mathbf{z_N} | \theta)|_{\theta=0}}{P_{\mathbf{Z_1}, \ldots, \mathbf{Z_N}}(\mathbf{z_1}, \ldots, \mathbf{z_N} | \theta = 0)} \tag{12}$$

where $P_{\mathbf{Z}_1,\ldots,\mathbf{Z}_N}(\mathbf{z}_1,\ldots,\mathbf{z}_N|\theta)$ is the joint p.d.f. of the matrices associated with each of the N sensors. The form of this test statistic will depend on the assumed model for the dependency between the signal observations at different sensors. We provide results for independent random signal observations at each sensor in part A of this section and for a common random signal model in part B.

A. Independent Observations

For the case of i.i.d S_i^p with the common p.d.f. f_S one finds that

$$P_{\mathbf{Z}_1,\ldots,\mathbf{Z}_N}(\mathbf{z}_1,\ldots,\mathbf{z}_N|\theta) = \prod_{p=1}^{N}\prod_{i=1}^{n}\prod_{j=1}^{M_p}\left(\int_{x_i^p=t_{j-1}^p}^{t_j^p}\int_{s_i^p=-\infty}^{\infty} f_S(s_i^p)f(x_i^p-\theta s_i^p)ds_i^p dx_i^p\right)^{z_{ij}^p} \tag{13}$$

and the LO test statistic from (12) becomes

$$\lambda_{LO} = \sum_{p=1}^{N}\sum_{i=1}^{n}\sum_{j=1}^{M_p} z_{ij}^p \ell_j^p \quad \text{where } \ell_j^p = \frac{f'(t_j^p)-f'(t_{j-1}^p)}{F(t_j^p)-F(t_{j-1}^p)} \tag{14}$$

where F is the cumulative distribution function corresponding to f. The result in (14) says that if the p^{th} sensor quantizer decides for the j^{th} symbol at discrete-time i then a factor of ℓ_j^p is added to λ_{LO}. From (14) it is easy to show that the efficacy of the LO scheme is the sum of the efficacies of the LO schemes obtained when each of the individual sensor quantizers are acting alone. Thus, we want to maximize the efficacies of each of these schemes based on each of the individual quantizers and this can be accomplished by using the single sensor theory discussed in Section II. In fact, if we require each sensor quantizer to use M levels, then the efficacy can be maximized by using identical copies of the best quantizer with M levels at each sensor which is found by using the theory of Section II. Since the results for the multiple sensor case are based on the single sensor results, in some cases the LO sensor quantizers will be nonsymmetric. An interesting feature is that if we require the whole group of sensor quantizers to use a fixed number of levels in any way they choose, it is sometimes best to have the quantizers each use a different number of levels. This represents a different kind of asymmetry from that previously discussed. As a specific example, we considered a case with two sensors and Gaussian noise where the sensors must use only a combined total of 8 levels. Using our results for the single sensor cases, we found that the scheme which uses the best 5-level quantizer and the best 3-level quantizer will perform better asymptotically than a scheme which uses two instances of the best 4-level quantizer. The efficacy of the scheme with different quantizers was 2.953 while the efficacy of the scheme with identical quantizers was 2.942. This difference is not large but it may be important for some applications. This phenomenon was also present for other generalized Gaussian noise cases with parameters 1.8 and 1.6 and the efficacy difference seemed to increase as the parameter decreased. For a generalized Gaussian parameter of 1.6, the scheme using the best 5 and a 3-level quantizers had an efficacy of 3.016 while the scheme using two of the best 4-level quantizers had an efficacy of 2.946.

B. Common Signal

The results for i.i.d. signal observations at each sensor are interesting, but these results are based heavily on single sensor results. In the common signal case the results do not rely on and are generally quite different from those for a single sensor. This is reasonable since the system being studied is essentially a distributed detection system with dependent observations from sensor to sensor and the best designs of such systems are often different from those for cases with independent observations [5]. For the common signal case where each signal sample S_i has the p.d.f. f_S, one finds

$$P_{\mathbf{Z}_1,\ldots,\mathbf{Z}_N}(\mathbf{z}_1,\ldots,\mathbf{z}_N|\theta) = \prod_{i=1}^{n} \int_{s_i=-\infty}^{\infty} f_S(s_i) \prod_{p=1}^{N} \prod_{j=1}^{M_p} \left(\int_{x_i^p=t_{j-1}^p}^{t_j^p} f(x_i^p - \theta s_i) dx_i^p \right)^{z_{ij}^p} ds_i \quad (15)$$

From (12) the LO test statistic is

$$\lambda_{LO} = \sum_{i=1}^{n} \left\{ \sum_{p=1}^{N} \sum_{j=1}^{M_p} z_{ij}^p \ell_j^p + \sum_{p=1}^{N} \sum_{j=1}^{M_p} \sum_{q=1,q\neq p}^{N} \sum_{r=1}^{M_r} z_{ij}^p \tilde{\ell}_j^p z_{ir}^q \tilde{\ell}_r^q \right\} \quad (16)$$

where ℓ_j^p is defined in (14) and

$$\tilde{\ell}_j^p = \frac{f(t_j^p) - f(t_{j-1}^p)}{F(t_j^p) - F(t_{j-1}^p)} \quad (17)$$

The terms in (16) have an interpretation similar to the interpretation given for those in (14). It is clear that the AO fusion scheme in this case is generally different from that for the independent signal case. The difference is the addition of a new term in the common signal LO fusion rule (second term of (16)). This new term is an attempt to form a correlation term similar to what one might form in centralized [4] and in distributed [5] detection problems. Using (4) the efficacy of a scheme which uses (16) can be shown to be

$$\xi = \sum_{p=1}^{N} \sum_{j=1}^{M_p} \ell_j^p (f'(t_j^p) - f'(t_{j-1}^p)) + \sum_{p=1}^{N} \sum_{j=1}^{M_p} \sum_{q=1,q\neq p}^{N} \sum_{r=1}^{M_r} \tilde{\ell}_j^p \tilde{\ell}_r^q (f(t_j^p) - f(t_{j-1}^p))(f(t_r^q) - f(t_{r-1}^q)) \quad (18)$$

A major simplification was accomplished in (18) by assuming both f and f' are zero at $\pm\infty$ which is often true. We have developed necessary conditions from (18) for an AO set of sensor thresholds similar to those given in (7) and (8). These $\prod_{p=1}^{N}(M_p - 1)$ equations are obtained by setting the derivatives of (18) with respect to the $t_j^p; p = 1, \ldots, N$ $j = 1, \ldots, M_p - 1$ equal to zero. The AO breakpoints can also be found by using a gradient search technique directly on (18) and we employed this technique for a case with two sensors ($N = 2$) and with binary quantizers ($M_1 = M_2 = 2$) at each sensor. Our results show that the AO quantizer breakpoints are different from those for the independent signal case in Part A. For the case of Gaussian noise, the AO quantizer for the common signal case used $t_1^1 = t_1^2 = 0.8255$ and due to symmetry considerations either (or both) of the thresholds could be negative with the same magnitude. For the case of independent signals the AO quantizer used $t_1^1 = t_1^2 = 1.5750$ where again the symmetry considerations apply. Similar results were found for cases with other generalized Gaussian p.d.f.s and these results are summarized in Table 1. The efficacies of the AO quantizer designs for the common random signal case are also given. From Table 1 it is clear that the optimum sensor quantizers for the case of two-levels are not even symmetric as we would expect.

IV. Conclusion

Some interesting new results on single and multisensor quantization for random signal detection have been presented. For single sensor cases we found that cases with an even number of levels usually produced asymptotically best quantizer characteristics which were nonsymmetric, while cases with an odd number of levels usually produced asymptotically best quantizer characteristic which were symmetric. The necessary conditions for the single sensor case were shown not to have an even symmetric solution for many cases with an even number of levels. These cases include all those where the LO nonlinearity is even symmetric and has a global minimum at zero. It even includes cases where the noise does not have a generalized Gaussian p.d.f. and the signal detection problem does not involve random signals. We also demonstrated through Monte Carlo simulations that our asymptotically best nonsymmetric quantizers perform better than the asymptotically best symmetric quantizers for some cases with a finite observation sample size and non-zero signal strengths.

For cases with multiple sensors and signal observations which are independent, it was demonstrated that the AO quantization scheme could be found using results from single sensor cases. It was shown that if the total group of sensor quantizers can use a fixed number of total levels in any manner they choose, it is sometimes better to use quantizers with a different number of levels at each sensor. For the case of a common random signal at multiple sensors, we found that the best quantizer designs and the method of fusing the quantized observations are generally much different from both the single sensor results and the results for independent signal observations. This difference occurs because the quantization and fusion schemes attempt to make use of the correlation between observations at different sensors.

Appendix A.

We now demonstrate analytically that an even symmetric quantizer solution with an even number of levels can not solve the necessary conditions for AO quantization given in (7) and (8) for any problem where the LO nonlinearity is even symmetric with a global minimum at zero. An even symmetric quantizer with an even number of levels must have a break point at $x = 0$ and the levels on each side of this breakpoint, call them q_- and q_+, must have the same value. Thus the necessary conditions involving these quantities (8) require

$$g_o(0) = \frac{(q_- + q_+)}{2} \tag{19}$$

which becomes

$$q_- = q_+ = g_o(0) \tag{20}$$

Now using the expression for the levels (7) yields

$$q_+ = \frac{\int_0^{t_+} g_o(x) f(x) dx}{\int_0^{t_+} f(x) dx} \tag{21}$$

where t_+ is the next closest break point to 0 in the positive direction. This can also be expressed as

$$\int_0^{t_+} [g_o(0) - g_o(x)] f(x) dx = 0 \tag{22}$$

If the LO nonlinearity has a global minimum at 0 then $g_o(0) - g_o(x) < 0 \ \forall x$. Since $f(x) > 0 \forall x$ the integrand is always negative and thus there is no solution for t_+.

References

[1] H. V. Poor and J. B. Thomas, "Optimum quantization for local decisions based on independent samples," *J. Franklin Inst.*, vol. 303, No. 1, pp. 549-561, Jan. 1977.

[2] H. V. Poor and D. Alexandrou, "A general relationship between two quantizer design criteria," *IEEE Transactions on Information Theory*, IT-26, No. 2, pp. 210-212, March 1980.

[3] E. Abaya and G. L. Wise, "Some Notes on Optimal Quantization," *IEEE International Conference on Communications*, pp. 30.7.1-30.7.5, June 1981.

[4] S. A. Kassam, *Signal Detection in Non-Gaussian Noise*, Springer-Verlag, New York, 1988.

[5] R. S. Blum and S. A. Kassam, "Locally optimum distributed detection with dependent sensors," *IEEE Transactions on Information Theory*, IT-38, No. 3, pp. 1066-1079, May 1992.

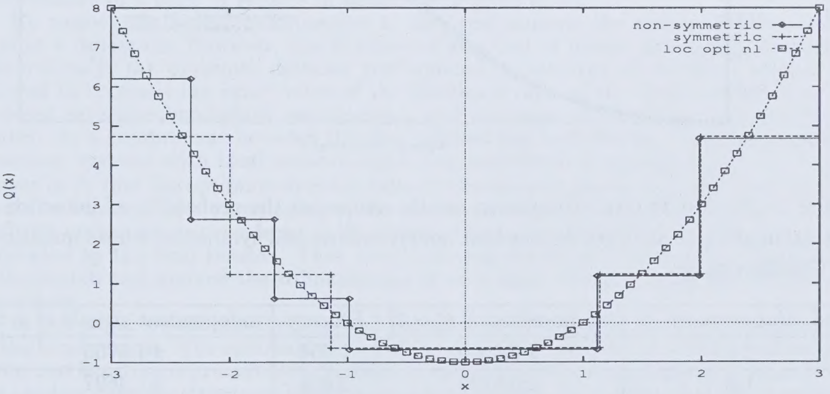


Figure 1: Our asymptotically best 5-level (dash) and 6-level (solid) quantizers with the locally optimum nonlinearity (dash and block) for a random signal in Gaussian noise.

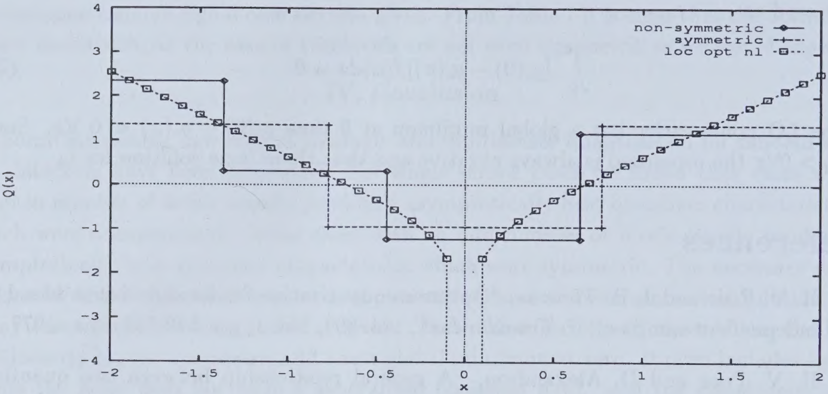


Figure 2: Our asymptotically best 3-level (dash) and 4-level (solid) quantizers with the locally optimum nonlinearity (dash and block) for a random signal in generalized Gaussian noise with $k = 1.6$.

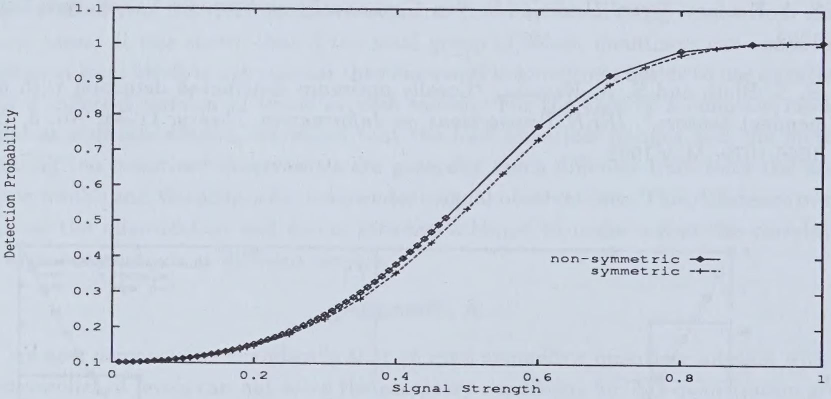


Figure 3: The Monte Carlo simulation results comparing the probability of detection as a function of signal strength for our best nonsymmetric and symmetric 4-level quantizers for Gaussian noise.

gen. Gauss. parm. k	common signal $t_1^1 = t_1^2$	Efficacy	independent signals $t_1^1 = t_1^2$
2.0	±0.8255	1.654	±1.5750
1.8	±0.0125	2.034	±1.3697
1.6	±0.0331	2.742	±1.0808

Table 1: Breakpoints and efficacies of asymptotically optimum quantizers for common random signal in generalized Gaussian noise with parameter k as compared to the breakpoints for independent signals.

Some Results on Distributed Detection Schemes Using Multilevel Quantization

Asis Nasipuri & Sawasd Tantaratana
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

Abstract

In this paper, we study distributed signal detection using two local sensors which generate M-level outputs, $M > 2$. Both the parallel and tandem structures are considered. A number of different nonrandomized fusion schemes are studied for the parallel network. These detectors require a higher channel bandwidth than those using binary quantization at the local sensors but they provide superior performances. The optimum quantizers at the local sensors are derived using the Bayes cost criterion.

I. Introduction

Distributed signal detection has been extensively studied for the case where the local sensor outputs are one bit (binary) hard decisions. The pioneering work in [1] solved the Bayesian detection problem for the network with two local sensors in parallel performing binary quantization of their individual observations. The optimum (binary) quantizers for a given fusion rule and cost function were derived. The problem of optimizing the fusion rule was considered in [2] when the local sensor decision rules are known, also from the point of view of minimizing a given cost function. These two concepts are combined in [3] to derive the global optimum solution of a two-sensor parallel network with fusion, assuming binary quantization at the local sensors. The tandem distributed detection structure has also received considerable attention [4]–[7]. In [4], it was demonstrated that a team consisting of two independent sensors in tandem achieves at least as good performance as the team consisting of two sensors in parallel. The optimum binary decision rules for a detection system with two sensors in tandem using the minimum cost criterion was presented in [5]. Performances of tandem detection networks were presented and compared with other known distributed detectors in [6]. The asymptotic performance of a team of sensors in tandem was given in [7].

By considering binary quantization at the local sensors, the communication cost is kept at a minimum. However, this is achieved at a cost of poorer detector performance. Theoretically, the optimum detector performance is achieved if the local sensors are allowed to transmit the exact value of the likelihood ratio of the observations. But this imposes extensive bandwidth requirements and increases the complexity of the fusion center. As a compromise between the two, interest has been drawn towards distributed detection systems with local sensors employing multilevel quantization [8], [9]. It was proved in [8] that the optimum decision rules of local sensors generating multilevel outputs would be achieved by a partitioning of the likelihood ratios of the individual observations. Multibit outputs were considered in [9], obtained by subpartitioning of the hard decisions generated by the local sensors. They used J–divergence as an objective function for the optimization and derived the subpartitions of each local sensor independent of those of the others.

In this paper we address the Bayesian detection problem using multilevel quantization at the local sensors. The optimum design of the quantizer functions in order to minimize a given cost function is presented in Sections II and III for the parallel and in Section IV for the tandem detector structures involving two local sensors. It is shown that the thresholds determining the optimum quantization functions can be solved from a set of coupled nonlinear equations. Numerical results are presented illustrating the performances of distributed detector networks with two local sensors using 2,3, and 4–level quantizations.

II. Parallel Network

The distributed parallel detection network with two local sensors D_1 and D_2, and a fusion center D_0 is illustrated in Figure 1(a). D_i receives an observation Y_i and produces the output U_i based on y_i, i.e., $u_i = q_i(y_i)$, where $q_i(\cdot)$ is the quantization function performed at D_i, $i = 1, 2$. We use upper case letters to denote random variables (r.v.'s) and lower case letters for their realizations. The fusion center generates the final decision U_0 based on $\{u_1, u_2\}$ according to some fusion rule $u_0 = g(u_1, u_2)$. The following assumptions are made on the system :

1. Y_i, $i = 1, 2$, belong to one of two distinct classes : H_0 (signal absent), or H_1 (signal present). Y_1 and Y_2 are independent r.v.'s with known probability density functions (pdf's) under each class, i.e.
$$f(y_1, y_2|H_j) \;=\; f(y_1|H_j)f(y_2|H_j) \tag{1}$$
for $j = 0, 1$, where $f(y_i|H_j)$ is known for $i = 1, 2$, $j = 0, 1$.
2. U_i is a discrete valued signal taking values in the set $\{0, 1, \cdots, M-1\}$ depending on Y_i only, $i = 1, 2$.
3. The output of the fusion center U_0 takes one of two values, 0 or 1, indicating acceptance of signal absent condition (H_0), or signal present condition (H_1), respectively. The fusion rule $g(u_1, u_2)$ is a function of u_1 and u_2 only.
4. There are costs associated with the two types of detection errors :
$$\text{Type 1}: \delta(u_0 = 1, H_0) = c_1 \qquad\qquad \text{Type 2}: \delta(u_0 = 0, H_1) = c_2 \tag{2}$$
where c_1 and c_2 are positive valued quantities known to the designer, and $\delta(u_0 = 0, H_0) = \delta(u_0 = 1, H_1) = 0$. Here, $\delta(u_0 = j, H_i)$ is the cost of accepting H_j when H_i is true.
5. The *a priori* probabilities of the classes, $P(H_0)$ and $P(H_1)$, are known.

With these assumptions we address the problem of determining the optimal local decision functions $q_i(\cdot)$, $i = 1, 2$ for a known fusion rule $g(\cdot, \cdot)$ so as to minimize the expected cost of the final decision, i.e.

$$\min_{q_i(\cdot),\ i=1,2} E[\delta(U_0, H)] \tag{3}$$

To solve this problem we follow the approach used in [1], and begin with the case of total decentralization without fusion. A cost function $J(u_1, u_2, H)$ based on the outputs of the local sensors and the true hypothesis H is defined. We give the solution for ternary quantization, i.e. $M = 3$, and show that the results can be easily extended to $M > 3$. The cost function $J(u_1, u_2, H)$ is defined as a mapping

$$J : \{0, 1, 2\} \times \{0, 1, 2\} \times \{H_0, H_1\} \longrightarrow \mathcal{R}$$

We further assume that

$$\begin{aligned} &J(i, u_2, H_0) \geq J(i-1, u_2, H_0), \quad J(i, u_2, H_1) \leq J(i-1, u_2, H_1) \\ &J(u_1, i, H_0) \geq J(u_1, i-1, H_0), \quad J(u_1, i, H_1) \leq J(u_1, i-1, H_1), \quad i = 1, 2 \end{aligned} \tag{4}$$

Our objective at this point is to minimize $E[J(u_1, u_2, H)]$ over all possible local decision rules. The function to be minimized can be written as

$$\begin{aligned} E[J(u_1, u_2, H)] &= \sum_{H, u_1, u_2} J(u_1, u_2, H)P(u_1, u_2, H) \\ &= \sum_{H, u_1, u_2} \int_{y_1, y_2} J(u_1, u_2, H)P(u_1|y_1)P(u_2|y_2)f(y_1|H)f(y_2|H)P(H) \end{aligned} \tag{5}$$

We look at the optimization of the decision functions of one local sensor at a time, under the assumption that the decision function of the other local sensor is fixed. To this end, we expand (5) as follows :

$$E[J(u_1,u_2,H)] \;=\; \int_{y_1} (I_0(y_1)P(u_1=0|y_1) + I_1(y_1)P(u_1=1|y_1) + I_2(y_1)P(u_1=2|y_1)) \tag{6}$$

where

$$\begin{aligned} I_0(y_1) &= \sum_{H,u_2} \int_{y_2} P(H)P(u_2|y_2)f(y_1|H)f(y_2|H)J(0,u_2,H) \\ I_1(y_1) &= \sum_{H,u_2} \int_{y_2} P(H)P(u_2|y_2)f(y_1|H)f(y_2|H)J(1,u_2,H) \\ I_2(y_1) &= \sum_{H,u_2} \int_{y_2} P(H)P(u_2|y_2)f(y_1|H)f(y_2|H)J(2,u_2,H) \end{aligned} \tag{7}$$

With these, the solution to the minimization with respect to y_1 is obtained as

$$\begin{array}{ll} \text{if } \; I_0(y_1) < I_1(y_1) \text{ and } I_2(y_1) & \text{choose } u_1 = 0 \\ \text{if } \; I_1(y_1) < I_2(y_1) \text{ and } I_0(y_1) & \text{choose } u_1 = 1 \\ \text{if } \; I_2(y_1) < I_0(y_1) \text{ and } I_1(y_1) & \text{choose } u_1 = 2 \end{array} \tag{8}$$

From (7) and (8) the decision regions for D_1 can be expressed in standard decision theory notation as:

$$\sum_{H,u_2} \int_{y_2} P(H)P(u_2|y_2)f(y_1|H)f(y_2|H)J(0,u_2,H) \; \underset{0 \text{ or } 2}{\overset{1 \text{ or } 2}{\gtrless}} \; \sum_{H,u_2} \int_{y_2} P(H)P(u_2|y_2)f(y_1|H)f(y_2|H)J(1,u_2,H) \tag{9}$$

$$\Longrightarrow \frac{f(y_1|H_1)}{f(y_1|H_0)} \underset{0 \text{ or } 2}{\overset{1 \text{ or } 2}{\gtrless}} \frac{\sum_{u_2} \int_{y_2} P(H_0)P(u_2|y_2)f(y_2|H_0)\{J(1,u_2,H_0) - J(0,u_2,H_0)\}}{\sum_{u_2} \int_{y_2} P(H_1)P(u_2|y_2)f(y_2|H_1)\{J(0,u_2,H_1) - J(1,u_2,H_1)\}} \tag{10}$$

Note that the left–hand side of (10) is the likelihood ratio of the observation at D_1. The right–hand side of (10), which we denote by τ_{01}, is fixed if the local decision function of D_2 is fixed. The other partitions of the likelihood ratio of Y_1 are obtained similarly as

$$\begin{aligned} \frac{f(y_1|H_1)}{f(y_1|H_0)} \underset{1 \text{ or } 0}{\overset{2 \text{ or } 0}{\gtrless}} & \frac{\sum_{u_2} \int_{y_2} P(H_0)P(u_2|y_2)f(y_2|H_0)\{J(2,u_2,H_0) - J(1,u_2,H_0)\}}{\sum_{u_2} \int_{y_2} P(H_1)P(u_2|y_2)f(y_2|H_1)\{J(1,u_2,H_1) - J(2,u_2,H_1)\}} \\ & \triangleq \tau_{11} \end{aligned} \tag{11}$$

and

$$\begin{aligned} \frac{f(y_1|H_1)}{f(y_1|H_0)} \underset{0 \text{ or } 1}{\overset{2 \text{ or } 1}{\gtrless}} & \frac{\sum_{u_2} \int_{y_2} P(H_0)P(u_2|y_2)f(y_2|H_0)\{J(2,u_2,H_0) - J(0,u_2,H_0)\}}{\sum_{u_2} \int_{y_2} P(H_1)P(u_2|y_2)f(y_2|H_1)\{J(0,u_2,H_1) - J(2,u_2,H_1)\}} \\ & \triangleq \tau_{21} \end{aligned} \tag{12}$$

This implies that the locally optimum solutions for D_1 is obtained as a partition of the likelihood ratio of the observation Y_1. Depending on the values of τ_{01}, τ_{11} and τ_{21} the decision space may look different. Two of the possible likelihood ratio space partitions are illustrated in Figure 2.

It is easy to see that the locally optimum decision function $q_2(u_2)$ can be obtained from (10), (11) and (12) by interchanging the roles of y_1 and y_2, u_1 and u_2, respectively. Hence, the corresponding thresholds describing the likelihood ratio partitions of Y_2 are given by

$$\tau_{02} = \frac{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_0)\{J(u_1,1,H_0)-J(u_1,0,H_0)\}P(H_0)}{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_1)\{J(u_1,0,H_1)-J(u_1,1,H_1)\}P(H_1)} \tag{13}$$

$$\tau_{12} = \frac{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_0)\{J(u_1,2,H_0)-J(u_1,1,H_0)\}P(H_0)}{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_1)\{J(u_1,1,H_1)-J(u_1,2,H_1)\}P(H_1)} \tag{14}$$

$$\tau_{22} = \frac{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_0)\{J(u_1,2,H_0)-J(u_1,0,H_0)\}P(H_0)}{\sum_{u_1}\int_{y_1} P(u_1|y_1)f(y_1|H_1)\{J(u_1,0,H_1)-J(u_1,2,H_1)\}P(H_1)} \tag{15}$$

Equations (10)–(15) can be solved simultaneously for the global solution. However, there might be multiple solutions, each being a locally optimum solution. The globally optimum solution is obtained by picking the best (lowest cost) solution among all the locally optimum solutions.

The above analysis can be easily extended to similar parallel networks with local sensors employing higher levels of quantization, i.e. $M > 3$. For example, for a decentralized detection system with $M = 4$, there are 4 thresholds to solve for each local sensor. Hence the global solution is obtained from the simultaneous solution of 8 nonlinear equations.

Example 1 : In this numerical example we assume that Y_1 and Y_2 are independent and identically distributed Gaussian r.v.'s with pdf's defined as

$$\begin{aligned} H_0 &: \; f(y_i|H_0) \sim \mathcal{N}(0;\sigma) \\ H_1 &: \; f(y_i|H_1) \sim \mathcal{N}(\theta;\sigma), \quad \theta > 0, \quad i = 1,2 \end{aligned} \tag{16}$$

where $\mathcal{N}(\theta;\sigma)$ denotes a Gaussian pdf with mean θ and variance σ^2. We assume the following cost function :

$$\begin{aligned} J(u_1,u_2,H_0) &= (u_1+u_2)^k \\ J(u_1,u_2,H_1) &= (4-(u_1+u_2))^k \end{aligned} \tag{17}$$

where k, $k \geq 1$, is a parameter determining the penalty for the quantizer outputs being far from the "expected" value. Intuitively, the expected values of the quantizer outputs under H_0 should be close to 0, and the outputs under H_1 closer to 2. Note that the definition in (17) is consistent with the assumptions in (4).

For a Gaussian r.v. Y_i with conditional pdf's as defined in (16), the test

$$\frac{f(y_i|H_1)}{f(y_i|H_0)} \overset{\text{A}}{\underset{\text{B}}{\gtrless}} \tau \tag{18}$$

can be equivalently expressed as

$$y_i \overset{\text{A}}{\underset{\text{B}}{\gtrless}} \frac{\theta}{2} + \frac{\log\tau}{\theta} \tag{19}$$

where A and B are the decisions corresponding to the likelihood ratio test with threshold τ. Hence, for each threshold τ_{i1} for D_1 and each threshold τ_{i2} for D_2, the corresponding thresholds for y_1 and y_2, are $t_{ij} = \theta/2 + \frac{\log\tau_{ij}}{\theta}$, $i = 0,1,2$, $j = 1,2$.

For the Gaussian pdf and with the cost function as defined in (17), it turns out that the locally optimum threshold τ_{2i} lies between τ_{0i} and τ_{1i}, $i = 1, 2$. As a matter of fact, $\tau_{21} = \tau_{22} = 1$, independent of k. Hence the decision regions for D_i are completely described by two thresholds τ_{0i} and τ_{1i}, as illustrated in Figure 2(a). Consequently, the number of equations we have to solve reduces to 4, given by

$$\tau_{01} = \frac{\int_{-\infty}^{t_{02}} f(y_2|H_0)A_0 + \int_{t_{02}}^{t_{12}} f(y_2|H_0)A_1 + \int_{t_{12}}^{\infty} f(y_2|H_0)A_2}{\int_{-\infty}^{t_{02}} f(y_2|H_1)B_0 + \int_{t_{02}}^{t_{12}} f(y_2|H_1)B_1 + \int_{t_{12}}^{\infty} f(y_2|H_1)B_2} \tag{20}$$

$$\tau_{11} = \frac{\int_{-\infty}^{t_{02}} f(y_2|H_0)E_0 + \int_{t_{02}}^{t_{12}} f(y_2|H_0)E_1 + \int_{t_{12}}^{\infty} f(y_2|H_0)E_2}{\int_{-\infty}^{t_{02}} f(y_2|H_1)F_0 + \int_{t_{02}}^{t_{12}} f(y_2|H_1)F_1 + \int_{t_{12}}^{\infty} f(y_2|H_1)F_2} \tag{21}$$

$$\tau_{02} = \frac{\int_{-\infty}^{t_{01}} f(y_1|H_0)A_0 + \int_{t_{01}}^{t_{11}} f(y_1|H_0)A_1 + \int_{t_{11}}^{\infty} f(y_1|H_0)A_2}{\int_{-\infty}^{t_{01}} f(y_1|H_1)B_0 + \int_{t_{01}}^{t_{11}} f(y_1|H_1)B_1 + \int_{t_{11}}^{\infty} f(y_1|H_1)B_2} \tag{22}$$

$$\tau_{12} = \frac{\int_{-\infty}^{t_{01}} f(y_1|H_0)E_0 + \int_{t_{01}}^{t_{11}} f(y_1|H_0)E_1 + \int_{t_{11}}^{\infty} f(y_1|H_0)E_2}{\int_{-\infty}^{t_{01}} f(y_1|H_1)F_0 + \int_{t_{01}}^{t_{11}} f(y_1|H_1)F_1 + \int_{t_{11}}^{\infty} f(y_1|H_1)F_2} \tag{23}$$

where the quantities A_i, B_i, E_i and F_i are defined as

$$\begin{aligned} A_i &= J(1,i,H_0) - J(0,i,H_0), & B_i &= J(0,i,H_1) - J(1,i,H_1) \\ E_i &= J(2,i,H_0) - J(1,i,H_0), & F_i &= J(1,i,H_1) - J(2,i,H_1) \end{aligned} \tag{24}$$

for each $i = 0, 1, 2$. Equations (20)–(23) were solved simultaneously for different values of k, and the solutions for $\theta = 1$, $\sigma = 1$ are illustrated in Figure 3. For $k = 1$ to $k = 3.6$, there is only one set of solutions, the same for both quantizers. However, for $k > 3.6$, 3 sets of solutions are obtained. For one set, the two local sensors have identical solutions, whereas for the other two sets they have different solutions. However, the solutions corresponding to the set of identical thresholds for both local sensors are found to have the minimum cost. The global solution for the entire range of $k = 0$ to 4 is shown by the solid lines, which represent solutions for both the local sensors.

III. Parallel Network With Fusion

When fusion of the local sensor outputs is considered, we need to minimize the expected value of the global cost function $E[\delta(U_0, H)]$. With the fusion rule expressed by $P(u_0|u_1, u_2)$, this problem can be placed in the previous framework by defining

$$J(u_1, u_2, H) = P(u_0 = 0|u_1, u_2)\delta(0, H) + P(u_0 = 1|u_1, u_2)\delta(1, H) \tag{25}$$

We are interested to obtain the receiver operating characteristics (ROC) of the optimum (using Bayes cost criterion) parallel two sensor detection system. This is done by defining

$$\begin{aligned} \delta(0, H_0) &= \delta(1, H_1) = 0 \\ \delta(1, H_0) &= 1 \\ \delta(0, H_1) &= a \qquad a \in (0, \infty) \end{aligned} \tag{26}$$

and obtaining the optimum local sensor quantizers for a given fusion rule. For the ternary quantization case, the following fusion rules are of interest :
Greater than 2 :

$$P(u_0 = 1|u_1, u_2) = \begin{cases} 1 & \text{if } u_1 + u_2 > 2 \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

Greater than 1 :

$$P(u_0 = 1|u_1, u_2) = \begin{cases} 1 & \text{if } u_1 + u_2 > 1 \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

The other possible fusion rules will not be as efficient, as they do not always use all the 3 distinct levels of the quantized signals from the local sensors. For example, the "Greater than 3" fusion rule makes no distinction between $u_1 = 0$ and $u_1 = 1$, as the final output of the fusion center is 0 for both of these outputs.

Once the fusion rule is fixed, the cost function $J(u_1, u_2, H)$ is defined by (25). The procedure of Section II can then be applied to obtain the optimum thresholds. The ROC function of the detection system is obtained by computing the probability of false alarm (P_{FA}) and probability of detection (P_{DET}) for different values of the parameter a in the range $(0, \infty)$ where P_{FA} and P_{DET} are given by

$$\begin{aligned} P_{FA} &= \sum_{u_1,u_2} \int_{y_1,y_2} P(u_0 = 1|u_1, u_2)P(u_1|y_1)P(u_2|y_2)f(y_1|H_0)f(y_2|H_0) && (29) \\ P_{DET} &= \sum_{u_1,u_2} \int_{y_1,y_2} P(u_0 = 1|u_1, u_2)P(u_1|y_1)P(u_2|y_2)f(y_1|H_1)f(y_2|H_1) && (30) \end{aligned}$$

IV. Tandem Network

Consider the detector structure shown in Figure 1(b). D_1 receives an observation Y_1 and generates a multilevel quantized output U_1 for the use of D_2, where $U_1 \in \{0, 1, 2, \cdots, M-1\}$. D_2 receives an observation Y_2, independent of Y_1, and, based on U_1 and Y_2, generates the ultimate decision U_2, $U_2 \in \{0, 1\}$. We address the problem of minimizing the expected cost $E[J(U_2, H)]$ over all possible local decisions u_1 and u_2, where $u_1 = q_1(y_1)$, $u_2 = q_2(y_2, u_1)$, and $J(\cdot, \cdot)$ is defined as the mapping

$$J : \{0, 1\} \times \{H_0, H_1\} \longrightarrow \mathcal{R} \tag{31}$$

The pdf's of Y_1 and Y_2 are assumed to be known for both signal present (H_1) and signal absent (H_0) cases. We derive the optimum threshold solutions for $M = 3$. Results for $M > 3$ easily follow from the same analysis. The mathematical approach used here is similar to that used for the parallel network. Assuming that $q_2(\cdot, \cdot)$ is fixed, we derive the optimal quantizer $q_1(\cdot)$ so as to minimize the expected cost. Next, the same is done for $q_2(\cdot, \cdot)$ assuming that $q_1(\cdot)$ is fixed. The global solution can be obtained from the simultaneous solution of the equations obtained from the above procedures. Expanding $E[J(U_2, H)]$ explicitly over all possible realizations of U_1 we obtain

$$\begin{aligned} E[J(U_2, H)] &= \sum_{u_2,H} J(u_2, H)P(u_2, H) \\ &= \sum_{i=0}^{2} \sum_{u_2,H} \int_{y_1,y_2} J(u_2, H)P(u_2|u_1 = i, y_2)P(u_1 = i|y_1) \\ &\qquad \cdot f(y_1|H)f(y_2|H)P(H) && (32) \end{aligned}$$

Define

$$\begin{aligned} I_0(y_1) &= \sum_{H,u_2} \int_{y_2} J(u_2, H)P(u_2|u_1 = 0, y_2)f(y_1|H)f(y_2|H)P(H) && (33) \\ I_1(y_1) &= \sum_{H,u_2} \int_{y_2} J(u_2, H)P(u_2|u_1 = 1, y_2)f(y_1|H)f(y_2|H)P(H) && (34) \\ I_2(y_1) &= \sum_{H,u_2} \int_{y_2} J(u_2, H)P(u_2|u_1 = 2, y_2)f(y_1|H)f(y_2|H)P(H) && (35) \end{aligned}$$

It follows that the optimum quantizer function $q_1(y_1)$ can be obtained from a similar procedure as shown in Section II, and is given by the partitioning of the likelihood ratio space of Y_1 by three thresholds τ_{01}, τ_{11} and τ_{21}, as follows

$$\Lambda(y_1) \underset{0 \text{ or } 2}{\overset{1 \text{ or } 2}{\gtrless}} \tau_{01} \qquad \Lambda(y_1) \underset{1 \text{ or } 0}{\overset{2 \text{ or } 0}{\gtrless}} \tau_{11} \qquad \Lambda(y_1) \underset{0 \text{ or } 1}{\overset{1 \text{ or } 2}{\gtrless}} \tau_{21} \tag{36}$$

where $\Lambda(y_1) = \frac{f(y_1|H_1)}{f(y_1|H_0)}$ is the likelihood ratio of y_1. The thresholds τ_{01}, τ_{11} and τ_{21} are derivable from the decision rule of D_2 and are explicitly given as

$$\begin{aligned}
\tau_{01} &= \frac{\sum_{u_2}\int_{y_2} J(u_2,H_0)[P(u_2|u_1=1,y_2)-P(u_2|u_1=0,y_2)]f(y_2|H_0)P(H_0)}{\sum_{u_2}\int_{y_2} J(u_2,H_1)[P(u_2|u_1=0,y_2)-P(u_2|u_1=1,y_2)]f(y_2|H_1)P(H_1)} \\
\tau_{11} &= \frac{\sum_{u_2}\int_{y_2} J(u_2,H_0)[P(u_2|u_1=2,y_2)-P(u_2|u_1=1,y_2)]f(y_2|H_0)P(H_0)}{\sum_{u_2}\int_{y_2} J(u_2,H_1)[P(u_2|u_1=1,y_2)-P(u_2|u_1=2,y_2)]f(y_2|H_1)P(H_1)} \\
\tau_{21} &= \frac{\sum_{u_2}\int_{y_2} J(u_2,H_0)[P(u_2|u_1=2,y_2)-P(u_2|u_1=0,y_2)]f(y_2|H_0)P(H_0)}{\sum_{u_2}\int_{y_2} J(u_2,H_1)[P(u_2|u_1=0,y_2)-P(u_2|u_1=2,y_2)]f(y_2|H_1)P(H_1)}
\end{aligned} \tag{37}$$

To derive the optimum $q_2(\cdot,\cdot)$, we expand the expected cost function (32) as follows.

$$\begin{aligned}
E[J(U_2,H)] = \sum_{i=0}^{2}\{\sum_{H}\int_{y_1,y_2} & [J(0,H)P(u_2=0|u_1=i,y_2) \\
& +J(1,H)P(u_2=1|u_1=i,y_2)]P(u_1=i|y_1)f(y_1|H)f(y_2|H)P(H)\}
\end{aligned} \tag{38}$$

This expression is minimized for each $u_1 = i$ by the following rule

$$\begin{aligned}
&\text{set } P(u_2=0|u_1,y_2)=1 \qquad \text{if} \qquad \sum_{H}\int_{y_1}\{J(0,H)-J(1,H)\}P(u_1=i|y_1) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot f(y_1|H)f(y_2|H)P(H)<0 \\
&\text{set } P(u_2=0|u_1,y_2)=0 \;\; \text{otherwise}
\end{aligned} \tag{39}$$

Relation (39) can be expressed in the following form

$$\frac{f(y_2|H_1)}{f(y_2|H_0)} \underset{u_2=0}{\overset{u_2=1}{\gtrless}} \frac{(J(1,H_0)-J(0,H_0))P(u_1=i|H_0)P(H_0)}{(J(0,H_1)-J(1,H_1))P(u_1=i|H_1)P(H_1)} \tag{40}$$

for $i = 0, 1, 2$, where

$$P(u_1=i|H) = \int_{y_1} P(u_1=i|y_1)f(y_1|H) \tag{41}$$

Therefore, the locally optimum quantizer function $q_2(y_2, u_1)$ is given by a likelihood ratio test, where the likelihood ratio $\Lambda(y_2)$ is compared to thresholds τ_{01}, τ_{11} or τ_{21}, depending on whether u_1 is 0,1, or 2, respectively, giving the final decision u_2, i.e.

$$\Lambda(y_2) \underset{u_2=0}{\overset{u_2=1}{\gtrless}} \tau_{i2} \;\; \text{when } u_1=i,\; i=0,1,2 \tag{42}$$

The optimum thresholds are dependent on the quantizer function of D_1, and is expressed by the RHS of (40). These results can be easily extended to higher number of quantization levels. The ROC of the tandem detection system can be obtained by defining $J(u_2, H)$ similar to $\delta(u_0, H)$ in (26). The P_{FA} and P_{DET} of the tandem network are given by

$$P_{FA} = \sum_{i=0}^{2} \int_{y_2} P(u_0 = 1 | u_1 = i, y_2) f(y_2|H_0) P(u_1 = i|H_0) \qquad (43)$$

$$P_{DET} = \sum_{i=0}^{2} \int_{y_2} P(u_0 = 1 | u_1 = i, y_2) f(y_2|H_1) P(u_1 = i|H_1) \qquad (44)$$

Example 2 : In this example we compare the performances of two-sensor distributed detection systems, for both parallel and tandem networks. We obtain the ROC's of 3 and 4-level quantized optimum detectors and compare them with those of the binary optimum structure and the optimum likelihood ratio detector without communication cost constraints. We assume that the observations at the two sensors are iid with a Gaussian pdf as described in Example 1. Results are plotted in Figure 4.

For the case of parallel network using binary quantization, two kinds of fusion are possible : AND (greater than 1) fusion, and OR (greater than 0) fusion. The quantizer function at each local sensor is described by a single threshold. Hence the optimization requires simultaneous solution of two nonlinear equations.

For $M = 3$, 6 simultaneous equations need to be solved for optimization. We compute the ROC's of systems employing the "Greater than 2" and the "Greater than 1" fusion rules.

For $M = 4$, again two types of fusion rules are worth studying. These are the "Greater than 3" and the "Greater than 2" fusion rules, defined the same way as before.

V. Conclusion

We have considered an extension of the Bayesian distributed signal detection problem to multilevel quantizations at the local sensors. Both the parallel network with fusion, and the tandem network, involving two local sensors are solved under this scheme. In comparison to binary quantization at the local sensors these detectors have lower error probabilities at the expense of a higher communication cost. Relative comparison shows that the performances of the detection networks improve with the number of quantization levels. Performance of the parallel network depends on the fusion rule employed, and no single fusion rule gives uniformly superior results for all values of the error probabilities. The performance of the tandem network is superior than that of the parallel network, as expected.

Acknowledgement

This work was supported by the National Science Foundation under Grant NCR-8821703.

References

[1] Tenney, R.R., and Sandell, N.R., "Detection With Distributed Sensors", *IEEE Trans. Aero. & Elect. Sys.*, AES-17, July 1981, pp. 501-510

[2] Chair, Z., and Varshney, P.K., "Optimum Data Fusion in Multiple Sensor Detection Systems", *IEEE Trans. Aero. & Elect. Sys.*, AES-22, Jan 1986, pp. 98-101.

[3] Reibman, A.R., and Nolte, L.W., "Optimal Detection and Performance of Distributed Sensor Systems", *IEEE Trans. Aero. & Elect. Sys.*, AES-23, Jan 1987, pp. 24-30

[4] Papastavrou, J.D., and Athans, M., "On Optimal Distributed Decision Architectures in a Hypothesis Testing Environment", 29th IEEE CDC, 1990.

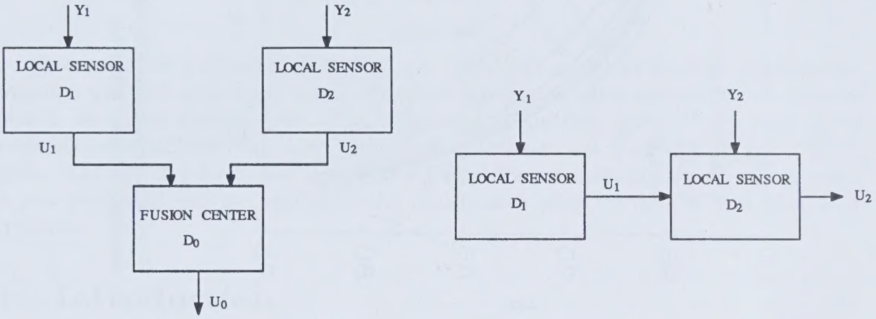
[5] Ekchian, L.K., and Tenney, R.R., "Detection Networks", 21st IEEE CDC, Dec 1982.

[6] Vishwanathan, R., Thomopoulos, S.C.A., and Tumulluri, R., "Optimal Serial Distributed Decision Fusion", *IEEE Trans. Aero. & Elect. Sys.*, AES-24, July 1988, pp. 366–375.

[7] Papastavrou, J.D., and Athans, M., "Distributed Detection by a Large Team of Sensors in Tandem", 29th IEEE CDC, 1990.

[8] Warren, D.J., and Willet, P.K., "Optimal Decentralized Detection for Conditionally Independent Sensors", American Control Conference, 1989.

[9] Lee, C.C., and Chao, J.J., "Optimal Local Decision Space Partitioning for Distributed Detection", *IEEE Trans. Aero. & Elect. Sys.*, AES-25, July 1989, pp. 536–543.


Y1
Y2
LOCAL SENSOR D1
LOCAL SENSOR D2
U1
U2
FUSION CENTER D0
U0
Y1
Y2
LOCAL SENSOR D1
U1
LOCAL SENSOR D2
U2

(a) Parallel network with fusion (b) Tandem network

Figure 1 : Distributed detection systems using two local sensors

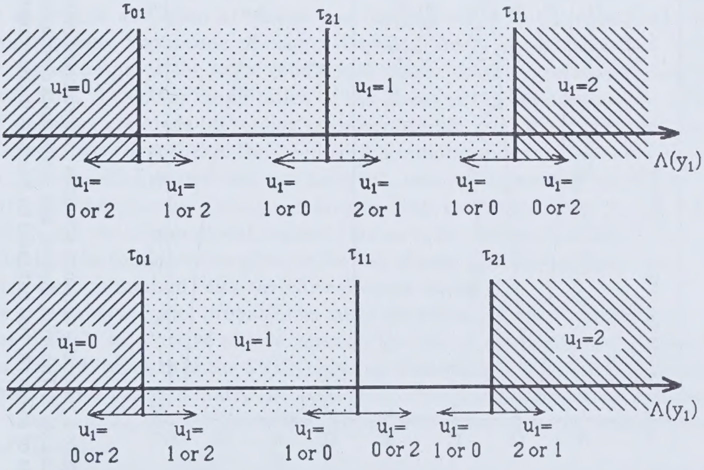

τ01
τ21
τ11
u1=0
u1=1
u1=2
Λ(y1)
u1= 0 or 2
u1= 1 or 2
u1= 1 or 0
u1= 2 or 1
u1= 1 or 0
u1= 0 or 2
τ01
τ11
τ21
u1=0
u1=1
u1=2
Λ(y1)
u1= 0 or 2
u1= 1 or 2
u1= 1 or 0
u1= 0 or 2
u1= 1 or 0
u1= 2 or 1

Figure 2 : Thresholds of the Likelihood ratio test for M=3

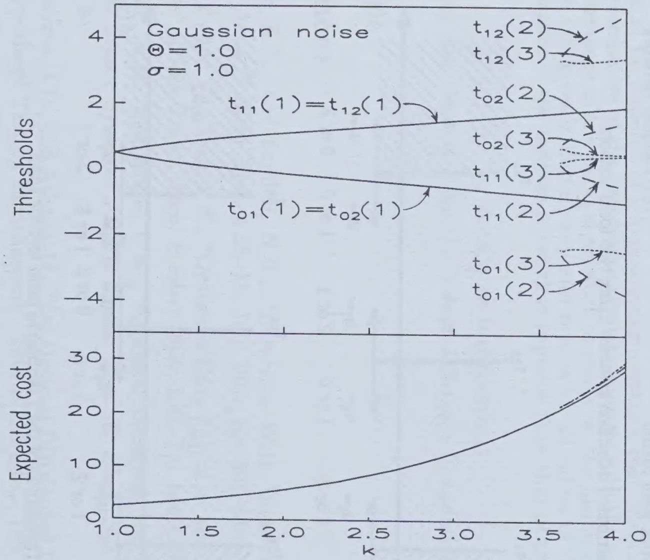


Figure 3 : Locally optimum solutions for thresholds obtained in Example 1. The solid, dashed, and dotted lines correspond to 3 different sets of solutions, also indicated by numbers inside parentheses.

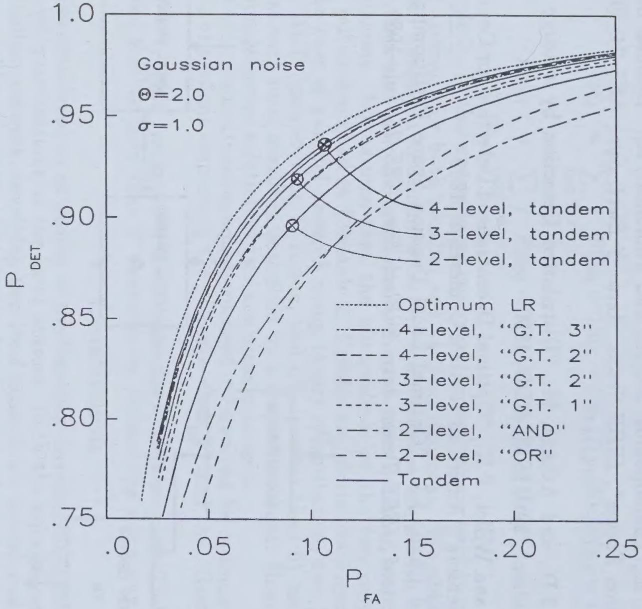


Figure 4 : Receiver operating characteristics of two-sensor distributed detector networks.

Classification using Set-Valued Kalman Filtering and Risk-Sensitive Bayes Theory

Scott E. Budge Todd K. Moon

September 28, 1992

Abstract

We consider the problem of using epistemic utility (risk-sensitive) decision theory for classification when the hypotheses are of different informational values using estimates obtained from a set-valued Kalman filter. The background of epistemic utility theory with convex probabilities is outlined and a brief introduction to set-valued Kalman filtering (SVKF)is given. The classifier method is applied to a multiple-target tracking/classification scenario. A new probability density, appropriate for classification using the ratio of intensities, is introduced.

1 Introduction

Bayes theory, for all its historical significance and mature theoretical development, continues to be re-examined with regards to the requirement for knowledge of *a priori* probability distributions. A fresh philosophical viewpoint has been recently promoted by Stirling and Morrell [1, 2, 3] based on the work of Levi [4, 5, 6]. In these works, two modifications are made to the theory. In the first place, a convex set of prior probabilities is postulated, eliminating the need for restrictive and often unrealistic assumptions about initial conditions. An elegant estimation-theoretic application of convex Bayesian theory is the set-valued Kalman filter (SVKF), whose output is the set of conditional means based on the observations and a convex set of priors. The other modification of the theory is the use of epistemic utility theory, in which decisions are made by trading off informational value vs. correctness. As in traditional Bayes theory, hypotheses are accepted if a criterion function exceeds some threshold. However, all exceeding hypotheses are accepted, leading to multiple decisions (suspended judgment) and the criterion function measures both the truth value (the so-called credal probability) and the informational content (the so-called information-determining probability). In light of the consideration given to the risk of error and informational value, we choose to call this decision method "risk-senstitive decision theory."

While convex sets of prior probabilities make sense from a decision-theoretic point of view, it may be difficult to conceive of how actually the sets of priors may be obtained in many problems. Indeed, other than the SVKF, the use of convex sets has seen little application. It is the intent of this paper to marry the ideas of set-valued Kalman filtering and risk-sensitive decision theory, using the output of the set-valued Kalman filter to form a convex set of likelihood functions.

The epistemic utility decision-making framework has been applied with some success to the problem of multiple-target tracking, where the tracks are represented by convex sets and the track-to-target association problem uses the epistemic utility theory [7]. The set-valued outputs provide a natural representation of regions of uncertainty in which association gating can be performed. This approach to multiple-target tracking provides automatic track initialization and reasonable handling of track crossings.

In our application of the decision theory, we will examine the problem of target classification. Data from the set-valued target estimates are incorporated into an function similar to a likelihood ratio for classification.

The remainder of this paper proceeds as follows. In Section 2 the salient features of convex Bayes decision theory are outlined. The fundamental ideas of the set-valued Kalman filter are briefly discussed in Section 3. In Section 4, the set-valued Kalman filter outputs are used to form sets of likelihood functions. In Section 5, an example of this joint estimation scheme in multiple-target tracking is presented.

2 Convex Bayes Theory

We present in this section a summary of the epistemological framework for decision making as originated by Levi [4] and propounded by Stirling and Morrell [1]. Under this epistemology, two measures are used in making decisions: the informational value of a hypothesis to the decision maker, and its truth value. For example, an agent, denoted by X, performing target classification would place much more informational value on a threat than on a benign target. A decision rule incorporating such a value would be reluctant to reject the hypothesis of a threat, even though the probability of a threat may be small.

To formalize this theory, let U denote the set of all hypotheses under consideration by X at a particular juncture. U is called the ultimate partition for X. Let n be the number of hypotheses to choose from in U. X may select any subset of hypotheses from U — he is not restricted to taking one and only one. As X may select various subsets of U, there are 2^n potential answers available for X. Let g denote a set of potential answers under consideration by $X, g \subset U$.

To obtain information, X must reject certain hypotheses. The alternative of accepting all hypotheses in U conveys no information to X — it is like selecting all choices in a multiple-choice exam when only one should be selected to impart information. On the other hand, accepting all hypotheses avoids error; certainly one of the choices must be true. There is thus a tradeoff between information and error.

By writing $U = \{h_1, h_2, \ldots, h_n\}$ and $g \subset U$, we define the informational utility of rejecting the subset g by $M(g)$, where

$$M(g) = \sum_{h_i \in g} M(h_i)$$

and $M(h_i)$ is the utility of rejecting a single hypothesis. The informational utility is constrained so that $M(h_j) > 0$ and $\sum_{j=1}^{n} M(h_j) = 1$; M is termed the information-determining probability. If $M(h_i) < M(h_j)$ then rejecting h_j is informationally more valuable than rejecting h_i; equivalently, accepting h_i is of more information value than accepting h_j.

The error-determining utility is represented by $Q(g)$, which is the so-called credal-probability. $Q(h_i)$ represents the agent's subjective probability (analogous to the prior probability in classical Bayesian reasoning) that h_i is true.

The tradeoff between information value and credal value can be written as the expected utility function. This utility is called the epistemic utility and can be written as [1]

$$u(g) = Q(g) - bM(g) \tag{1}$$

where b is a parameter used to weight the importance of error-avoidance versus informational value to the agent X. The parameter b is termed the boldness, $0 \leq b \leq 1$. If $b = 1$, then the informational value M is weighed heavily in comparison to avoidance of error (as determined by the credal probability Q). Classical Bayesian methods with uniform cost of errors corresponds to $b = 0$. X's best policy for decision making, given his credal probability Q and his information-determining probability M, is to maximize $u(g)$. This can be accomplished if the agent takes all and only those $h_i \in g$ such that

$$u(h_i) = Q(h_i) - bM(h_i) > 0.$$

Reorganizing this, agent X accepts h_i iff

$$Q(h_i)/M(h_i) > b.$$

This ratio test is similar to ratio tests common in traditional Bayes theory. In the context of the classification problem discussed below, it will be called the classification likelihood ratio test (CLRT).

Further flexibility is introduced to the epistemology by the use of convex sets of credal probabilities. Suppose that agent X has two credal probabilities, Q and Q', and that X has compelling external reasons for accepting both of them and is thus unable to decide between them. This represents a state of ignorance for X. This ignorance, the inability to select a definite credal probability, is in contrast to uncertainty in the decision-making processing, which is already modeled by the density functions. Rather than forcing X to make an arbitrary decision of one particular density, one approach is to relax the requirement of a single credal probability and consider instead convex combinations of Q and Q', e.g.

$$\alpha Q + (1 - \alpha)Q'$$

for $\alpha \in [0, 1]$. This combination is called a convex credal combination. A set $\mathcal{B}_{X,t}$ denotes the credal state of X at time t. If $Q \in \mathcal{B}_{X,t}$ and $Q' \in \mathcal{B}_{X,t}$, then all convex combinations $Q^\alpha = \alpha Q + (1 - \alpha)Q' \in \mathcal{B}_{X,t}$ for all $\alpha \in [0, 1]$. As shown by Stirling [1], convex sets of credal probabilities are closed under conditioning. It is also possible, though not useful to us in the current context, to define convex sets of information-valuation functions.

3 Set-valued filtering

Related to the idea of convex sets of credal probabilities is the concept of set-valued estimation. Traditional Bayesian estimation propagates a density by means of conditional probability. The best-known instance of this is the Kalman filter, in which the mean and variance of a Gaussian density are propagated. One of the problems with traditional Bayesian estimation is the choice of the prior probabilities. Often priors must be chosen on the basis of subjective judgment. The improper selection of a prior results in biased or incorrect results for all stages of the estimation (although asymptotically the effect of the prior probability may vanish). Ignorance concerning the prior probability can be explicitly displayed using a convex set of priors and employing a set-valued estimation procedure. In set-valued estimation, rather than a single density, a whole family of densities is propagated. Since convex sets of credal probabilities are closed under conditioning, an initial convex set of densities is propagated as a set of densities under all conditional updates.

When applied to convex sets of Gaussian densities, the set-valued Kalman filter (SVKF) is produced. Rather than a point-valued mean, the set-valued Kalman filter propagates a convex set of conditional means. Rather elegantly, this set of means can be propagated in closed form, with a computational load only slightly greater than the point-valued Kalman filter [2]. This is rather remarkable, since the effect is the same as propagating an infinite number of Kalman filters, each with a different initial mean. A single common covariance matrix is associated with each mean to determine the Gaussian densities. To contrast the traditional (point-valued) Kalman filter with the SVKF, the sets produced by the set-valued

Kalman filter give a spread of first moments, whereas second-moment (covariance) computed is a measure of the spread of the true value relative to the estimate (only one first moment). Thus, the SVKF provides two measures of uncertainty that may be useful, rather than just one.

As the SVKF runs, the set of estimates converges asymptotically to a point estimate, provided that the system is observable. Lack of observability in a state variable leads to failure of convergence. The SVKF is thus well-suited to poorly observable systems. Those state variables which are observable will converge; those which are not will not. The size of the set is also an indication of the convergence of the filter. This may be more meaningful than examination of the posterior covariance matrix which is used in many cases to determine quality of estimate. Another viewpoint of the SVKF is that it provides all estimates that are consistent with the data, subject to the uncertainty of the initial condition. As more data are received, the initial uncertainty decreases and the sets decrease in size. Figure 3 illustrates the performance of the SVKF (and the association logic derived from the decision theory below) in a multiple-target tracking problem.

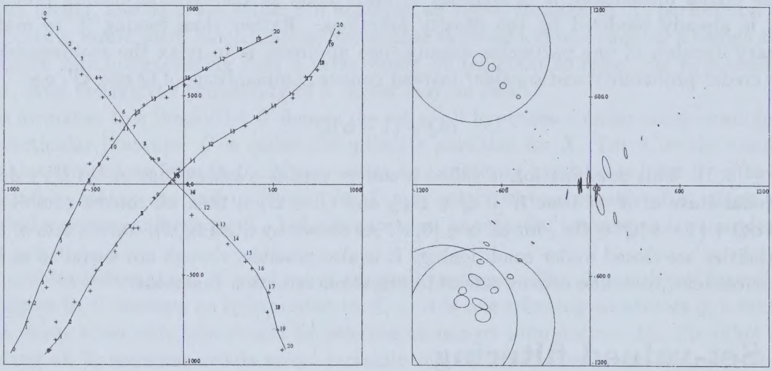


Figure 1: Three Crossing Tracks: (a) simulated trajectories and observations; (b) filtered track sets. (From [7]).

In what follows, we will use the SVKF as the estimator. The updated set of means at time t, using the measurement at time t will be denoted by $\underline{X}_{t|t}$, and is given by [2]

$$\underline{X}_{t|t} = \{\underline{x} \in \Re^n : ||K_{tt}^{-1}(\underline{x} - \underline{c}_{tt})|| \leq 1\}.$$

The set of means is an ellipsoid in n dimensions, n being the number of state variables. $\underline{c}_{tt}$ is the centroid of the ellipse of means and K_{tt} defines the extent of the ellipse about the centroid. The norm may be any convenient norm (we will use the L_2 norm).

4 Convex epistemic classification

In this section we will combine the use of the convex epistemic decision theory of Section 2 and the set-valued Kalman filter of Section 3 in a decision-theoretic application. At time t an agent X has a sequence of observations from τ_t targets and desires to assign each of the τ_t targets to one of T target types. Let the observations at time t be denoted by $\underline{z}_{it}$, $i = 1, 2, \ldots, s_t$, where s_t is the number of observations. For simplicity in the following discussion, it is assumed that the number of targets τ_t is equal to the number of observations s_t. (This is for clarity of exposition with respect to the classification issue. In many circumstances, e.g. ballistic tracking, the number of targets actually present may be more or less than the number of observations due to target clustering, clutter, data dropout, etc.) From each observation, a convex set of means is formed using the set-valued Kalman filter. Let these sets of means be denoted $\underline{X}_{it}, i = 1, 2, \ldots, s_t$.

Let the classification hypotheses by represented by h_{ijt}, where h_{ijt} is the hypothesis that target i (from observation i), $i = 1, 2, \ldots, s_t$, classifies as target type j, $j = 1, 2, \ldots, T$ at time t. Then for each target there is an ultimate partition

$$U_{it} = \{h_{i1t}, h_{i2t}, \ldots, h_{iTt}\}$$

The ith target is classified as being of type j if we fail to reject the hypothesis h_{ijt}.

Traditionally, decisions about classification hypotheses would be made strictly on the basis of the observed data. That is, we would form a likelihood function

$$L(h_{ijt}|\underline{z}_{it}) \tag{2}$$

and make our decision based on the likelihood function and the prior probability of each hypothesis. We are motivated to look beyond this traditional scheme for two reasons. First, the prior probabilities of the hypotheses may be unknown; indeed, it is not the probability of occurrence of a particular target type that may be of most interest to us but rather the importance of the target type. Second, in the presence of system and observation noise, we should be reluctant to use the simple observation $\underline{z}_{it}$ as the conditioning quantity in our likelihood function. Since we are classifying the target based on a history of observations, the decision should, if possible, be based on the entire history (including prior assumptions), not just the most recent noisy observation. The convex set of means $\underline{X}_{it}$ incorporates information from all observations. The uncertainty due to a single observation is thus averaged out. In addition, the set explicitly displays, by its size, all uncertainty introduced by the uncertainty in the initial assumptions.

In light of this reasoning, it makes sense to consider an epistemic utility approach. The credal probability function is determined upon the set-valued estimate, and we write it as $Q(h_{ijt}|\underline{X}_{it})$. By so doing, we have introduced a credal probability which is no longer unique. For each point $\underline{x} \in \underline{X}_{it}$, we have a different credal probability which we will write as $Q(h_{ijt}|\underline{x})$, each of which is consistent with all observations and prior assumptions. This set of credal probabilities is not convex, in the sense defined by Stirling. However, it is defined over a convex set of parameters (the set $\underline{X}_{it}$).

By using the estimated sets $\underline{X}_{it}$ as the basis for our decision making, we are performing something similar to decision-directed estimation [8], but in reverse. In this case we are using estimates (which are actually decision directed themselves) to form likelihoods to be used in a decision process. Such involution makes claims of optimality difficult. The merit of the method, however, is the ability to explicitly indicate what is known about the data.

To complete the epistemic utility formulation, to each hypothesis we assign a utility of rejection, and denote that utility by $M(h_{ijt}|\underline{X}_{it})$. The decision rule can be written conceptually as

$$Q(h_{ijt}|\underline{X}_{it}) > bM(h_{ijt}|\underline{X}_{it})$$

where b is a boldness parameter as described above. $Q(h_{ijt}|\underline{X}_{it})$ actually represents a whole family of credal probabilities and $M(h_{ijt}|\underline{X}_{it})$ actually represents a whole family of information-determining probabilities. Well-defined comparisons must be made at specific points $\underline{x} \in \underline{X}_{it}$. We note four possible alternative strategies for making this decision:

1. Accept any hypothesis such that
$$Q(h_{ijt}|\underline{x}) > bM(h_{ijt}|\underline{x}) \text{ for some } \underline{x} \in \underline{X}_{it}.$$
From a theoretical point of view, this is most defensible, as all points in $\underline{X}_{it}$ are equally possible according to the set-valued Kalman filter. However, from a computational point of view it may be hard to determine the existence of such a point $\underline{x}$.
2. Accept any hypothesis such that
$$\max_{\underline{x}\in\underline{X}_{it}} Q(h_{ijt}|\underline{x}) > b \min_{\underline{x}\in\underline{X}_{it}} M(h_{itj}|\underline{x}).$$
This strategy is reluctant to reject any hypotheses. It is the most cautious method, and would have the lowest probability of error, but gains information the slowest. Computationally, it involves both a constrained minimization and maximization, but these can often be performed using a gradient search method.
3. Accept any hypothesis such that
$$\min_{\underline{x}\in\underline{X}_{it}} Q(h_{ijt}|\underline{x}) > b \max_{\underline{x}\in\underline{X}_{it}} M(h_{itj}|\underline{x})$$
This is the boldest strategy, acquiring information (in the Levi sense) fastest. The error would, on the other hand, be the highest of the strategies. Computationally, it is about the same as strategy 2.
4. Select any hypothesis such that
$$\max_{\underline{x}\in\underline{X}_{it}} Q(h_{ijt}|\underline{x}) > bM(h_{itj}|\underline{x}).$$
In other words, find the point $\underline{x} \in \underline{X}_{it}$ that maximizes Q, and evaluate M at that point. This is still quite cautious, but avoids the step of another constrained optimization problem.

5 Application to multiple-target classification

In this section, the ideas presented previously are applied to a multiple-target classification problem. At every time t, s_t observations in C different intensity spectral wavebands are made and associated with τ_t track sets representing the set of possible means from a set-valued Kalman filter. The observed values are position (azimuth and bearing) and intensity, giving a three-dimensional observation vector. The details of the tracking, dynamics, and association are presented in [7]. The tracks from the different intensity spectral wavebands are combined together. A track set consists of a set of means representing all possible values of the state variables (position, velocities, and each different intensity) consistent with the observations and prior assumptions.

Classification of targets in this system is performed using ratios of intensities. To make the example more complete, assume that there are three spectral bands, denoted as color C_1, color C_2, and color C_3. The set of colors will be written $\mathcal{C}$. The intensities in each of these bands are written as I_{C1}, I_{C2}, and I_{C3}. The classification ratios used are $\rho_1 = I_{C_1}/I_{C_2}$ and $\rho_2 = I_{C_3}/I_{C_2}$. These two ratios will be written as an ordered pair $\boldsymbol{\rho} = (\rho_1, \rho_2) \in \Re^{C-1}$. In general with C colors, the set of ratios form a $(C-1)$-tuple.

It is assumed that there is a set of prior target ratios used for classification, denoted by $\mathbf{r}_j = (r_{1j}, r_{2j}), j = 1, 2, \ldots, T$. Classification will be made based on the similarity of the observed ratios $\boldsymbol{\rho}$ to the target ratios $\mathbf{r}$.

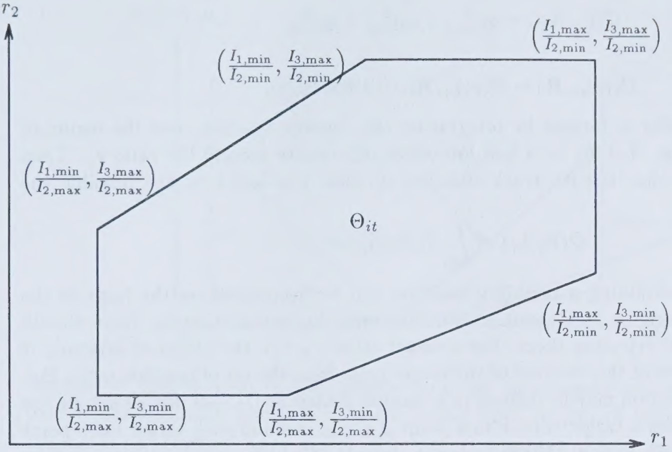

r_2
$\left(\frac{I_{1,\max}}{I_{2,\min}}, \frac{I_{3,\max}}{I_{2,\min}}\right)$
$\left(\frac{I_{1,\min}}{I_{2,\min}}, \frac{I_{3,\max}}{I_{2,\min}}\right)$
$\left(\frac{I_{1,\min}}{I_{2,\max}}, \frac{I_{3,\max}}{I_{2,\max}}\right)$
Θ_{it}
$\left(\frac{I_{1,\max}}{I_{2,\min}}, \frac{I_{3,\min}}{I_{2,\min}}\right)$
$\left(\frac{I_{1,\min}}{I_{2,\max}}, \frac{I_{3,\min}}{I_{2,\max}}\right)$
$\left(\frac{I_{1,\max}}{I_{2,\max}}, \frac{I_{3,\min}}{I_{2,\max}}\right)$
r_1

Figure 2: Set in (r_1, r_2) plane formed by ratios of means.

The "observations" of the intensities are taken from the set-valued Kalman filter. The intensities are thus assumed to have a Gaussian density (justifying the use of the Kalman filter) with mean $\underline{i}_{ict}$ and variance σ^2_{ict}, where the mean $\underline{i}_{it}$ comes from the set computed by the set-valued Kalman filter

$$\underline{i}_{ict} \in [I_{ict,\min}, I_{ict,\max}] \stackrel{\triangle}{=} \underline{I}_{ict}$$

for $c \in C$ and $i = 1, 2, \ldots, s_t$. The set of ratios of means consistent will all observations forms an irregular hexagon, as illustrated in Figure 2. Any point in this region is a valid ratio of means. This region of possible ratios for the ith target is denoted by the symbol $\Theta_{it} \subset \Re^{C-1}$ and the set of possible intensities for the ith target is denoted by $\mathcal{I}_{it} \subset \Re^C$. Any point in the set $\mathcal{I}_{it}$ has a unique corresponding point in Θ_i; the inverse mapping from Θ_{it} to $\mathcal{I}_{it}$ is not well defined for most elements in Θ_{it}.

A credal probability function is formed from the density function of a ratio non-zero mean Gaussian densities. The derivation of this density, a non-central Cauchy density, is not given here due to lack of space. The density can be written as

$$f_{\mathbf{r}}(\mathbf{r}|\underline{\mathbf{i}}_{it}) = \exp[(D^2 - C)/2] \frac{1}{2\pi\sqrt{|\mathbf{P}_{it}|A^3}} [1 + D^2]$$

where $\mathbf{r} = [r_1, r_2]'$, $\underline{\mathbf{i}}_{it} = [\underline{i}_{ic_1t}, \underline{i}_{ic_2,t}, \underline{i}_{ic_3t}]' \in \mathcal{I}_i$. $\mathbf{P}_{it}$ is the covariance matrix of the intensity estimate,

$$\mathbf{P}_{it} = \mathrm{diag}[\sigma^2_{ic_1t}, \sigma^2_{ic_2t}, \sigma^2_{ic_3t}]$$

where it assumed that the intensity estimates are all independent, and $\mathbf{R}_{it}$ is the inverse of $\mathbf{P}_{it}$:

$$\begin{aligned} \mathbf{R}_{it} &= \mathrm{diag}[1/\sigma^2_{ic_1t}, 1/\sigma^2_{ic_2t}, 1/\sigma^2_{ic_3t}] \\ &\stackrel{\triangle}{=} \mathrm{diag}[q_1, q_2, q_3] \end{aligned}$$

The quantities $A = A(\mathbf{r}; \mathbf{R}_{it})$, $B = B(\mathbf{r}; \underline{\mathbf{i}}_{it}, \mathbf{R}_{it})$, $C = C(\underline{\mathbf{i}}_{it}, \mathbf{R}_{it})$ and $D = D(\mathbf{r}; \underline{\mathbf{i}}_{it}, \mathbf{R}_{it})$ are determined by

$$A^2(\mathbf{r}; \mathbf{R}_{it}) = q_1 r_1^2 + q_6 r_2^2 + q_4$$

$$B(\mathbf{r};\underline{\mathbf{i}}_{it},\mathbf{R}_{it}) = -(2q_1\underline{i}_{ic_1t}r_1 + 2q_4\underline{i}_{ic_2t} + 2q_6\underline{i}_{ic_3t}r_2)$$

$$C(\underline{\mathbf{i}}_{it},\mathbf{R}_{it}) = q_1\underline{i}^2_{ic_1t} + q_4\underline{i}^2_{ic_2t} + q_6\underline{i}^2_{ic_3t}$$

and

$$D(\mathbf{r};\underline{\mathbf{i}}_{it},\mathbf{R}) = B(\mathbf{r};\underline{\mathbf{i}}_{it},\mathbf{R}_{it})/(2A(\mathbf{r};\mathbf{R}_{it})).$$

The credal probability is formed by integrating this density function over the region of interest for classification. Let B_j be a ball formed in ratio space around the ratio $\mathbf{r}_j$. Then the credal probability that the ith track classifies to class j is based on the possible the intensity $\underline{\mathbf{i}}_{it} \in \mathcal{I}_{it}$ is

$$Q(B_j|\underline{\mathbf{i}}_{it}) = \int_{B_j} f_{\mathbf{r}_j}(\mathbf{r}_j|\underline{\mathbf{i}}_{it})d\mathbf{r}.$$

An information-determining probability function can be formulated on the basis of the importance of the target, as mentioned above. For more important targets, there should be greater utility in not rejecting them. For a target ratio (r_1, r_2), the utility of rejecting it should also be a function of the distance of the target ratio from the set of possible ratios Θ_{it}. A reasonable utility function may be defined in a manner similar to [7]. Let $d(\mathbf{r},\boldsymbol{\rho})$ denote the (Euclidian) distance from a target ratio $\mathbf{r}$ to a point $\boldsymbol{\rho} \in \Theta_{it}$. Around each target ratio point define a seriously possible region, that is, a region outside of which no classification can be reasonably obtained. Denote the seriously possible region by Σ_{jt}. Now let the importance of a class be indicated by weighting the distance of the class target ratio from the set of possible mean ratios, Θ_{it}, normalized so that utility of rejection forms a probability. Let $m_{ijt}(\mathbf{r}_j,\boldsymbol{\rho}_{it})$ represent the information-determining probability of rejecting the hypothesis that target i is of class j at time t. Then we can write

$$m_{ijt}(\mathbf{r}_j,\boldsymbol{\rho}_{it}) = \begin{cases} b_j d(\mathbf{r}_j,\boldsymbol{\rho}_{it})/N_m & \boldsymbol{\rho}_{it} \in \Sigma_{jt} \\ \text{undefined} & \boldsymbol{\rho}_{it} \notin \Sigma_{jt} \end{cases}$$

where the normalizing factor N_m is such that

$$\sum_{j=1}^{T} \int_{\Sigma_{jt}} m_{ijt}(\mathbf{r}_j,\boldsymbol{\rho}_{it})d\mathbf{r} = 1.$$

The weighting function b_j is set so that there is less utility in rejecting important classes. Thus, more important classes have a smaller b_j. An illustration of this information-determining probability density is illustrated in Figure 5. The assignment of the seriously possible regions Σ_{jt} are part of the design problem and should reflect the agent's values of the targets.

The information value of rejecting the ball B_j based on the intensity $\boldsymbol{\rho}_{it} \in \Theta_{it}$ is

$$M_{it}(B_j|\boldsymbol{\rho}_{it}) = \int_{B_j\cap\Sigma_{jt}} m_{ijt}(\mathbf{r}_j,\boldsymbol{\rho}_{it})d\mathbf{r}$$

The credal probability and the information-determining probability can now be combined in the classification likelihood ratio test (CLRT). We will state this using strategy 4 from above. Let b be a fixed boldness. Then the CLRT accepts class j if

$$\frac{\max_{\underline{\mathbf{i}}_{it}\in\mathcal{I}_{it}} Q(B_j|\underline{\mathbf{i}}_{it})}{M_{it}(B_j|\boldsymbol{\rho}_{it})} > b.$$

By taking the ball around $\mathbf{r}_j$ infinitessimally small and using the continuity of the integrands, this CLRT can be written as

$$\frac{\max_{\underline{\mathbf{i}}_{it}\in\mathcal{I}_{it}} f(\mathbf{r}_j|\underline{\mathbf{i}}_{it})}{m_{ijt}(\mathbf{r}_j,\boldsymbol{\rho}_{it})} > b$$

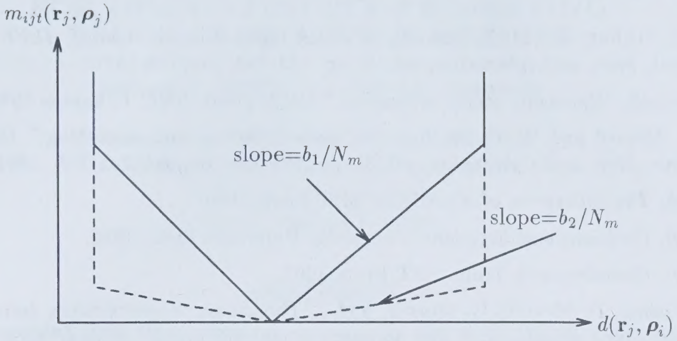

$m_{ijt}(\mathbf{r}_j, \boldsymbol{\rho}_j)$
slope=b_1/N_m
slope=b_2/N_m
$d(\mathbf{r}_j, \boldsymbol{\rho}_i)$

Figure 3: Illustration of the utility function for targets of two different priorities and equal seriously possible regions. Solid line: utility of rejecting a target of low importance. Dashed line: utility of rejecting a target of high importance.

This formulations requires computation of

$$\max_{\underline{\mathbf{i}}_{it} \in \mathcal{I}_{it}} f_{\mathbf{r}_j}(\mathbf{r}_j | \underline{\mathbf{i}}_{it}).$$

This is a constrained optimization problem, but is not too difficult due to the linear constraints and the fact that all derivatives can be readily computed. Taking the logarithm of the density function results in a function that is nearly quadratic to that points near the maximum can be obtained in one Newton iteration.

6 Discussion and conclusions

The use of the set-valued Kalman filter as the observation in the likelihood function has several features that recommend it. In the first place, except in the case of high noise observations, the set-valued estimate will usually include the observation point. The use of the set is thus (usually) a generalization of the epistemic utility, which in turn is a generalization of Bayesian methods. In those circumstances where the set does not contain the observation, it is because information preceding the observation suggests that the observation is noisy. The filtered data thus avoid incorrect classification due to a single bad observation.

In addition, the approach described above should provide some robustness with respect to the set of prior target ratios. The ratios $\mathbf{r}_j$ determine how targets are classified. However, it is unlikely that they will be known precisely. The use of the set-valued estimate to compare with the ratios combined with a maximization routine should make it so the system is somewhat forgiving of prior target ratios that are slightly incorrect. Further robustness could be provided using a set of prior target ratios consistent with all believed values of the prior target ratios.

The use of the noncentral Cauchy distribution for classification of ratio intensities also appears to be new here. Further investigation into properties of the noncentral Cauchy are necessary to determine the merit, if any, of this distribution to others such as, for example, the ratio of the squares of the intensities. It does, however, have a convenient closed form expression.

References

[1] W. C. Stirling and D. R. Morrell, "Convex bayes decision theory," *IEEE Trans. on Systems, Man, and cybernetics*, vol. 21, pp. 173–183, Jan/Feb 1991.

[2] D. Morrell, "Epistemic utility estimation," *IEEE Trans. SMC*, to appear 1993.

[3] D. R. Morrell and W. C. Stirling, "Set-valued filtering and smoothing," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, pp. 184–193, Jan/Feb Jan/Feb. 1991.

[4] I. Levi, *The Enterprise of Knowledge.* MIT Press, 1980.

[5] I. Levi, *Decisions and Revisions.* Cambridge University Press, 1984.

[6] I. Levi, *Gambling with Truth.* MIT Press, 1967.

[7] W. Stirling, D. Morrell, G. Morrell, and J. Thompson, "Convex bayes decision theory and set-valued estimation: A new approach to multiple target tracking," tech. rep., Final Report to ESL, Brigham Young University, 1991.

[8] D. Middleton and R. Esposito, "Simultaneous optimum detection and estimation of signals in noise," *IEEE Transactions on Information Theory*, vol. IT-14, no. 3, pp. 434–444, May 1968.

PARAMETER ESTIMATION FOR DISCONTINUITY-PRESERVING STOCHASTIC SIGNAL MODELS

Richard R. Schultz and Robert L. Stevenson

Laboratory for Image and Signal Analysis
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

Smoothing noisy data using a quadratic cost function tends to oversmooth edges. A discontinuity-preserving objective function for the data can be used to alleviate this problem, but the alternative functions that are available generally contain free parameters which must be estimated so that an optimal estimate of the data can be found. A method of estimating stochastic signal model parameter values is proposed which uses a Bayesian maximum *a posteriori* (MAP) estimation technique. In this paper, Bayesian estimation will be examined for selecting discontinuity-preserving function parameters and the temperature parameter of the Gibbs density directly from the observed data. Experiments will be conducted on a 1D data set, using both Bayesian parameter estimation and the cross-validation method.

1 INTRODUCTION

One of the most common problems in signal processing is that of extracting a signal which has been degraded and corrupted by noise. MAP estimation [2, 4, 8–10, 13] and maximum entropy [5, 11] techniques have been successful in recovering the data from the observations. However, estimation methods that utilize Gaussian models tend to oversmooth the data at discontinuities. Much recent research has investigated discontinuity-preserving signal processing [1–3, 10, 12]; that is, to extract the signal while maintaining the sharp edges and other distinguishing features from the original data. In this section, Bayesian MAP signal estimation will reviewed, and several discontinuity-preserving objective functions will be presented which may be incorporated into a statistical signal model.

1.1 Bayesian Signal Estimation

The system model which will be assumed for the observed data is

$$\mathbf{y} = \mathbf{H}\mathbf{z} + \mathbf{n}, \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^M$ is the observed signal data, $\mathbf{H} \in \mathbb{R}^{M \times N}$ is a linear degradation matrix operating on the uncorrupted signal $\mathbf{z} \in \mathbb{R}^N$, and $\mathbf{n} \in \mathbb{R}^M$ is additive noise. It will be assumed that the degradation matrix is known explicitly. The inverse problem will be investigated in this paper; that is, to find an estimate of the signal $\mathbf{z}$, given only the observed data $\mathbf{y}$.

The inverse problem described is often ill-posed, and a technique of reformulating the problem is through maximum *a posteriori* (MAP) estimation. To make the problem well-posed, assume a signal model $\mathcal{M}$ with some free parameters designated by θ. The MAP estimate is then defined as the maximum of the log-likelihood function,

$$\hat{\mathbf{z}} \;=\; \arg \max_{\mathbf{z} \in \mathbb{R}^N} \log \Pr(\mathbf{z}|\mathbf{y}, \theta, \mathcal{M}). \tag{2}$$

The posterior probability can be expanded by Bayes' rule, giving

$$\Pr(\mathbf{z}|\mathbf{y}, \theta, \mathcal{M}) \;=\; \frac{\Pr(\mathbf{y}|\mathbf{z}, \theta, \mathcal{M}) \Pr(\mathbf{z}|\theta, \mathcal{M})}{\Pr(\mathbf{y}|\theta, \mathcal{M})}. \tag{3}$$

This probability consists of a noise model $\Pr(\mathbf{y}|\mathbf{z}, \theta, \mathcal{M}) = \Pr(\mathbf{n})|_{\mathbf{n}=\mathbf{y}-\mathbf{Hz}}$ and a prior model for the data $\Pr(\mathbf{z}|\theta, \mathcal{M})$. The term in the denominator is not a function of $\mathbf{z}$, and therefore it serves as a normalizing constant. The noise model represents the additive noise corrupting the data, and it has the assumed density

$$\Pr(\mathbf{y}|\mathbf{z}, \theta, \mathcal{M}) \;=\; \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{C}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{y} - \mathbf{Hz})^t \mathbf{C}^{-1} (\mathbf{y} - \mathbf{Hz})\right\}, \tag{4}$$

representing zero-mean Gaussian noise with covariance matrix $\mathbf{C}$. A useful prior model is the Gibbs density

$$\Pr(\mathbf{z}|\theta, \mathcal{M}) \;=\; \frac{1}{Z} \exp\left\{-\frac{1}{\lambda} \sum_{c \in \mathcal{C}} V_c(\mathbf{z}, \alpha)\right\}, \tag{5}$$

where Z is known as the partition function, λ represents the "temperature" parameter, $V_c(\mathbf{z}, \alpha)$ is a function of a local group of points c called cliques, and $\mathcal{C}$ represents the set of all cliques throughout the signal [4]. The exponential argument $\Omega[\mathbf{z}, \theta] = \frac{1}{\lambda} \sum_{c \in \mathcal{C}} V_c(\mathbf{z}, \alpha)$ serves as a measure of consistency with respect to assumed data properties, dependent upon the parameters $\theta = [\alpha \;\; \lambda]^t$. For the class of signal models considered in this paper, the form of $\Omega[\mathbf{z}, \theta]$ will be

$$\Omega[\mathbf{z}, \theta] \;=\; \frac{1}{\lambda} \sum_{c \in \mathcal{C}} \rho_\alpha\left(\mathbf{d}_c^t \mathbf{z}\right), \tag{6}$$

in which $\rho_\alpha(\cdot)$ is an objective function dependent upon α, and $\mathbf{d}_c \in \mathbb{R}^N$ is a constant vector applied to the signal $\mathbf{z}$. An *a priori* assumption regarding signal properties is made (*e.g.*, global signal smoothness), and the measure of how consistently the data follows the assumption is given by the scalar $\mathbf{d}_c^t \mathbf{z}$. The objective function $\rho_\alpha(\mathbf{d}_c^t \mathbf{z})$ is used to allow some inconsistencies in the signal (*e.g.*, discontinuities). The prior and noise densities are used in equation (2), producing the MAP estimate of the data

$$\hat{\mathbf{z}} \;=\; \arg \min_{\mathbf{z} \in \mathbb{R}^N} \left\{\Omega[\mathbf{z}, \theta] + \frac{1}{2}(\mathbf{y} - \mathbf{Hz})^t \mathbf{C}^{-1} (\mathbf{y} - \mathbf{Hz})\right\}. \tag{7}$$

Many objective functions are available which can be used to allow inconsistencies in the signal model.

1.2 Objective Functions

The most typical selection for the objective function $\rho_\alpha(\cdot)$ is the quadratic function. However, using a quadratic objective function may result in an estimate $\hat{\mathbf{z}}$ which does not allow for inconsistencies to the prior signal assumptions. Many nonquadratic functions are available which may be used to allow inconsistencies to form in the signal estimate.

Nonconvex functions have been used by several researchers [1, 3, 7]. These functions have the disadvantage of resulting in nonconvex optimization problems, which must be solved with computationally expensive techniques such as simulated annealing and continuation methods to locate the estimate $\hat{\mathbf{z}}$. Blake and Zisserman [1] used the "neighbor interaction function," which is quadratic up to some threshold and constant beyond that value, *i.e.*,

$$\rho_\alpha(x) \;=\; \begin{cases} x^2, & |x| \le \alpha, \\ \alpha^2, & |x| > \alpha. \end{cases} \tag{8}$$

Li [7] proposed the irregularity function for a class of adaptive regularizers,

$$\rho_\alpha(x) \;=\; -\exp\left\{-\frac{x^2}{\alpha}\right\}. \tag{9}$$

More recently, Geman and Reynolds [3] used the function

$$\rho_\alpha(x) \;=\; -\frac{1}{1+\left|\frac{x}{\alpha}\right|} \tag{10}$$

for constructing prior densities of the data.

To alleviate the difficulties associated with nonconvex optimization, several convex objective functions have also been introduced. These functions provide a single global minimum for locating the MAP estimate in equation (7), and hence employ less expensive gradient-based techniques for forming the signal estimate. The generalized absolute function

$$\rho_\alpha(x) = |x|^\alpha, \qquad 1 \le \alpha \le 2, \tag{11}$$

was used by Bouman and Sauer [2] for tomographic image reconstruction. The Huber function was researched by Stevenson and Delp [12], which is quadratic up to some threshold and linear beyond that point. Explicitly, this function has the form

$$\rho_\alpha(x) \;=\; \begin{cases} x^2, & |x| \le \alpha, \\ \alpha^2 + 2\alpha(|x| - \alpha), & |x| > \alpha. \end{cases} \tag{12}$$

All the functions mentioned contain a single threshold, shape, or scaling parameter which must be estimated, dependent upon the observed data.

1.3 Overview

This paper will examine Bayesian parameter estimation for the objective function parameter α and the Gibbs density temperature parameter λ. An approximation will be used for the exact prior density utilizing a multivariate Taylor series expansion, so that a practical minimization can be derived for estimating the parameters. Experiments will select the parameters for a 1D data set, comparing the Bayesian parameter estimation technique to the cross-validation method.

2 BAYESIAN PARAMETER ESTIMATION

Finding the optimal parameters $\hat{\theta} = [\hat{\alpha}\ \hat{\lambda}]^t$ given only the observed signal data $\mathbf{y}$ and the assumed model $\mathcal{M}$ is a level of preprocessing, calculated previous to finding the Bayesian signal estimate $\hat{\mathbf{z}}$ [8, 9, 13]. This section will define the problem of Bayesian parameter estimation, describing the approximations which allow for the practical computation of $\hat{\theta}$.

2.1 Problem Statement

The Bayesian MAP parameter estimate is the maximum of the log-likelihood function

$$\hat{\theta} \;=\; \arg\max_{\theta \in \mathbf{R}^2} \log \Pr(\theta | \mathbf{y}, \mathcal{M}). \tag{13}$$

The posterior probability of the parameters can be expanded, producing

$$\Pr(\theta | \mathbf{y}, \mathcal{M}) \;=\; \frac{\Pr(\mathbf{y} | \theta, \mathcal{M}) \Pr(\theta | \mathcal{M})}{\Pr(\mathbf{y} | \mathcal{M})}, \tag{14}$$

so that the parameter estimate has the form

$$\hat{\theta} \;=\; \arg\max_{\theta \in \mathbf{R}^2} \left\{ \log \Pr(\mathbf{y} | \theta, \mathcal{M}) + \log \Pr(\theta | \mathcal{M}) - \log \Pr(\mathbf{y} | \mathcal{M}) \right\}. \tag{15}$$

The last term in equation (15) is not dependent upon the parameters θ, and it can be eliminated from the optimization as a constant. As for the parameter values, little is known *a priori* except that they must be positive. A uniform density will therefore be assumed for the parameter prior,

$$\Pr(\theta | \mathcal{M}) \;=\; \begin{cases} a, & \theta > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

The selected density allows for the logarithm of the prior to be eliminated from the optimization as a constant. This results in the constrained maximum-likelihood estimate of the parameters,

$$\hat{\theta} \;=\; \arg\max_{\theta \in \Theta} \log \Pr(\mathbf{y} | \theta, \mathcal{M}), \tag{17}$$

where $\Theta = \{\theta \,:\, \theta > 0\}$ is the set of all possible parameters.

The probability of having observed the data is given by the continuous total probability theorem, which states

$$\Pr(\mathbf{y} | \theta, \mathcal{M}) \;=\; \int_{\mathbf{z} \in \mathbf{R}^N} \Pr(\mathbf{y} | \mathbf{z}, \theta, \mathcal{M}) \Pr(\mathbf{z} | \theta, \mathcal{M})\, dz. \tag{18}$$

This expression can be expanded by using the statistical models for the noise and the signal,

$$\Pr(\mathbf{y} | \theta, \mathcal{M}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{C}|^{\frac{1}{2}} Z} \int_{-\infty}^{+\infty} \exp\left\{ -\Omega[\mathbf{z}, \theta] - \frac{1}{2} (\mathbf{y} - \mathbf{H}\mathbf{z})^t \mathbf{C}^{-1} (\mathbf{y} - \mathbf{H}\mathbf{z}) \right\} dz. \tag{19}$$

Consequently, the MAP parameter estimate becomes

$$\hat{\theta} = \arg\max_{\theta \in \Theta} \log \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{C}|^{\frac{1}{2}} Z} \int_{-\infty}^{+\infty} \exp\left\{ -\Omega[\mathbf{z}, \theta] - \frac{1}{2} (\mathbf{y} - \mathbf{H}\mathbf{z})^t \mathbf{C}^{-1} (\mathbf{y} - \mathbf{H}\mathbf{z}) \right\} dz. \tag{20}$$

The exact integral and the partition function Z are difficult to calculate. In order to form a computable estimate for $\hat{\theta}$, an approximation will be made to the data prior $\Pr(\mathbf{z} | \theta, \mathcal{M})$ to approximate both Z and the integral in equation (20).

2.2 Prior Density Approximation

To make the observation probability $\Pr(\mathbf{y}|\theta, \mathcal{M})$ calculable, a Taylor series approximation will be made for the exponential argument of the data prior $\Pr(\mathbf{z}|\theta, \mathcal{M})$. The first three terms of the multidimensional Taylor series expansion are

$$\Omega[\mathbf{z}, \theta] \approx \frac{1}{2}(\mathbf{z} - \mathbf{z}_0)^t \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta](\mathbf{z} - \mathbf{z}_0) + (\mathbf{z} - \mathbf{z}_0)^t \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta] + \Omega[\mathbf{z}_0, \theta]. \tag{21}$$

The Hessian matrix is denoted as $\nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta] \in \mathbb{R}^{N \times N}$ evaluated at $\mathbf{z}_0$,

$$\nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta] \;=\; \frac{1}{\lambda} \sum_{c \in C} \rho_\alpha''(\mathbf{d}_c^t \mathbf{z}) \mathbf{d}_c \mathbf{d}_c^t \bigg|_{\mathbf{z}=\mathbf{z}_0}. \tag{22}$$

In this expression, the second derivative of the function $\rho_\alpha(x)$ with respect to x is designated by $\rho_\alpha''(x)$. Likewise, $\nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta] \in \mathbb{R}^N$ is the gradient vector evaluated at $\mathbf{z}_0$,

$$\nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta] \;=\; \frac{1}{\lambda} \sum_{c \in C} \rho_\alpha'(\mathbf{d}_c^t \mathbf{z}) \mathbf{d}_c \bigg|_{\mathbf{z}=\mathbf{z}_0}. \tag{23}$$

An alternative form of the approximate $\Omega[\mathbf{z}, \theta]$ can be derived by matrix algebra manipulations, and it is given by the expression

$$\begin{aligned} \Omega[\mathbf{z}, \theta] \;\approx\; & \frac{1}{2}\left[\mathbf{z} - (\mathbf{z}_0 - \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta]^{-1} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta])\right]^t \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta] \\ & \times \left[\mathbf{z} - (\mathbf{z}_0 - \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta]^{-1} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta])\right] \\ & - \frac{1}{2} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta]^t \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta]^{-1} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta] + \Omega[\mathbf{z}_0, \theta]. \end{aligned} \tag{24}$$

Although both approximate forms of $\Omega[\mathbf{z}, \theta]$ are equivalent, the second expression in equation (24) is more useful for calculating the partition function Z of the density since it has the form of a Gaussian kernel. Using this Gaussian approximation, the partition function Z can be approximated as

$$Z = \frac{(2\pi)^{\frac{N}{2}}}{|\nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta]|^{\frac{1}{2}}} \exp\left\{\frac{1}{2} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta]^t \nabla_{\mathbf{z}}^2 \Omega[\mathbf{z}_0, \theta]^{-1} \nabla_{\mathbf{z}} \Omega[\mathbf{z}_0, \theta] - \Omega[\mathbf{z}_0, \theta]\right\}. \tag{25}$$

2.3 Integral Approximation

The integral in equation (20) will be reformulated to allow for the practical calculation of the parameter estimate. The essential idea used in this paper is to approximate the observation probability with a Gaussian form as the integral argument,

$$\Pr(\mathbf{y}|\theta, \mathcal{M}) \approx \frac{(2\pi)^{\frac{N}{2}} |\mathbf{B}|^{\frac{1}{2}} e^k}{(2\pi)^{\frac{N}{2}} |\mathbf{C}|^{\frac{1}{2}} Z} \int_{-\infty}^{+\infty} \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{B}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{z} - \mathbf{z}_{\hat{\theta}})^t \mathbf{B}^{-1} (\mathbf{z} - \mathbf{z}_{\hat{\theta}})\right\} dz, \tag{26}$$

with covariance matrix $\mathbf{B}$ and mean $\mathbf{z}_{\hat{\theta}}$. If this approximation is used, the observation probability reduces to the expression

$$\Pr(\mathbf{y}|\theta, \mathcal{M}) \;\approx\; \frac{e^k |\mathbf{B}|^{\frac{1}{2}}}{Z |\mathbf{C}|^{\frac{1}{2}}}. \tag{27}$$

By using the approximation for $\Omega[\mathbf{z},\theta]$ in equation (24) and the partition function Z in equation (25), the two exponential arguments from equation (20) can be combined as a sum of quadratic terms. Matrix algebra manipulations produce the form of the covariance matrix

$$\mathbf{B} \;=\; \left(\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\mathbf{H}^t\mathbf{C}^{-1}\mathbf{H}\right)^{-1} \tag{28}$$

and the constant term

$$\begin{aligned} k \;=\; & -\frac{1}{2}(\mathbf{z}_\theta-\mathbf{z}_0)^t\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta](\mathbf{z}_\theta-\mathbf{z}_0)-(\mathbf{z}_\theta-\mathbf{z}_0)^t\nabla_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]-\Omega[\mathbf{z}_0,\theta] \\ & -\frac{1}{2}(\mathbf{y}-\mathbf{H}\mathbf{z}_\theta)^t\mathbf{C}^{-1}(\mathbf{y}-\mathbf{H}\mathbf{z}_\theta). \end{aligned} \tag{29}$$

The mean $\mathbf{z}_\theta$ is the MAP signal estimate, with the approximate analytical form

$$\mathbf{z}_\theta \;=\; \left(\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\mathbf{H}^t\mathbf{C}^{-1}\mathbf{H}\right)^{-1}\left(\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]\mathbf{z}_0-\nabla_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\mathbf{H}^t\mathbf{C}^{-1}\mathbf{y}\right). \tag{30}$$

2.4 Parameter Optimization Problem

Using the approximations, the parameter estimate now has the form

$$\hat{\theta} \;=\; \arg\max_{\theta\in\Theta}\,\log\frac{e^k|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]|^{\frac{1}{2}}}{(2\pi)^{\frac{N}{2}}|\mathbf{C}|^{\frac{1}{2}}|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\mathbf{H}^t\mathbf{C}^{-1}\mathbf{H}|^{\frac{1}{2}}}, \tag{31}$$

which can be expressed as the minimization

$$\begin{aligned} \hat{\theta} \;=\; \arg\min_{\theta\in\Theta}\Big\{ & \frac{1}{2}\log\left|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\mathbf{H}^t\mathbf{C}^{-1}\mathbf{H}\right|-\frac{1}{2}\log\left|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]\right|+\frac{1}{2}\log|\mathbf{C}| \\ & +\frac{1}{2}(\mathbf{z}_\theta-\mathbf{z}_0)^t\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta](\mathbf{z}_\theta-\mathbf{z}_0)+(\mathbf{z}_\theta-\mathbf{z}_0)^t\nabla_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta]+\Omega[\mathbf{z}_0,\theta] \\ & +\frac{1}{2}(\mathbf{y}-\mathbf{H}\mathbf{z}_\theta)^t\mathbf{C}^{-1}(\mathbf{y}-\mathbf{H}\mathbf{z}_\theta)+\frac{N}{2}\log(2\pi)\Big\}. \end{aligned} \tag{32}$$

Search techniques can be used to locate the estimate.

3 PARAMETER ESTIMATION BY CROSS-VALIDATION

The technique of parameter estimation by cross-validation (CV) will be used as a comparison method to the proposed Bayesian technique. Cross-validation involves minimizing a function of prediction error to select the correct parameters [6,14,15]. $\mathbf{z}^k_\theta$ will be used to represent the signal estimate obtained for equation (7), with the k^{th} data point y_k left out of the error term.

Since $\mathbf{z}^k_\theta$ should be a relatively good estimate for the original data $\mathbf{z}$, then $\left[\mathbf{H}\mathbf{z}^k_\theta\right]_k$ should also be a good approximation to the missing data point y_k. The cross-validation criterion that will be used is the sum of squared prediction errors

$$CV(\theta) \;=\; \frac{1}{N}\sum_{k=0}^{N-1}\left(y_k-\left[\mathbf{H}\mathbf{z}^k_\theta\right]_k\right)^2. \tag{33}$$

The parameter estimate $\hat{\theta}$ is thus considered optimal for the minimum value of $CV(\theta)$.

4 EXPERIMENTS

The Huber function has been selected for use in the signal model. This objective function has been used successfully for maintaining the discontinuities within processed data [10, 12]. The convex Huber function is defined in Section 1.2, and it has a free parameter α which is a threshold separating the piecewise quadratic and linear sections.

An assumption of piecewise smoothness will be made for the signal data. This prior information can be included in the Gibbs density signal model by making $\mathbf{d}_c^t\mathbf{z}$ a smoothness measure. Typically, this involves calculating the first or second-order derivatives at each element within the signal. For the experiments, a finite difference approximation to the second-order derivative at element z_i will be used,

$$\mathbf{d}_i^t\mathbf{z} \;=\; z_{i-1} - 2z_i + z_{i+1}. \tag{34}$$

The experiments will select the Huber function threshold α and the temperature parameter λ of the Gibbs density, given only a noisy 1D data set and the described model selection. The mean squared error (MSE) and the mean absolute error (MAE) criteria are used to quantitatively compare the estimates produced by Bayesian estimation to those generated by CV.

In the case of data smoothing, the linear degradation operator of the system model in equation (1) is simply the $N \times N$ identity matrix, $\mathbf{H} = \mathbf{I}$. The additive noise is assumed to be i.i.d. Gaussian, with the diagonal noise covariance matrix $\mathbf{C} = \sigma^2\mathbf{I}$. The parameter estimate for white Gaussian noise smoothing becomes

$$\begin{aligned}\hat{\theta} \;=\; \arg\min_{\theta\in\Theta} \Big\{ & \frac{1}{2}\log\left|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z},\theta] + \frac{1}{\sigma^2}\mathbf{I}\right| - \frac{1}{2}\log\left|\nabla^2_{\mathbf{z}}\Omega[\mathbf{z},\theta]\right| + \frac{N}{2}\log\sigma^2 \\ & + \frac{1}{2}(\mathbf{z}_\theta - \mathbf{z}_0)^t\nabla^2_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta](\mathbf{z}_\theta - \mathbf{z}_0) + (\mathbf{z}_\theta - \mathbf{z}_0)^t\nabla_{\mathbf{z}}\Omega[\mathbf{z}_0,\theta] + \Omega[\mathbf{z}_0,\theta] \\ & + \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{z}_\theta\|^2 \Big\},\end{aligned} \tag{35}$$

in which $\frac{N}{2}\log(2\pi)$ has been eliminated from the minimization as a constant term.

Some practical implementation issues should be discussed for calculating the estimate of the parameters α and λ. Currently, a global search in the parameter space Θ is used locate $\hat{\theta}$. A sparse uniform sampling of the parameter space is searched, since the estimates within a neighborhood of Θ are very similar. Another item of note is that the optimization appears to be rather robust to the selection of $\mathbf{z}_0$. An arbitrary selection of the parameters $\alpha_0 = 100$ and $\lambda_0 = 100$ was used to generate $\mathbf{z}_0$ from the observed data in each of the Bayesian estimation experiments.

In the first experiment, white Gaussian noise of standard deviation $\sigma = 10$ was added to the original signal $\mathbf{z}$ in Figure 1. The parameter estimates using both the Bayesian technique and CV were calculated, and their corresponding signal estimates appear in Figure 2. The '×' symbol represents the observed signal data $\mathbf{y}$, and the solid line represents the estimate $\hat{\mathbf{z}}$ in the figures. Figure 2(a) shows the Bayesian signal estimate for $\hat{\alpha} = 82$ and $\hat{\lambda} = 1700$ overlaid upon the noisy observations. The data is smoothed slightly, and the discontinuities still remain. The CV estimate essentially remains consistent with the observed data, with very little smoothing taking place. The CV estimate is shown in Figure 2(b), for the parameters $\hat{\alpha} = 1$ and $\hat{\lambda} = 200$. These two estimates are compared quantitatively in Table 1. The MSE and the MAE values were calculated by comparing each estimated signal $\hat{\mathbf{z}}$ to the uncorrupted signal $\mathbf{z}$. The Bayesian estimate has lower error values, due to the small amount of smoothing allowed by the selected parameters. The CV estimate fits the observed data very closely, and therefore does not conform to the original signal as well as the MAP estimate.

In the second experiment, Gaussian noise of standard deviation $\sigma = 20$ was added to the data points. This results in a rather messy observed signal, from which the original signal must be extracted. Figure 3(a) shows the Bayesian estimate for $\hat{\alpha} = 103$ and $\hat{\lambda} = 800$ superimposed on the noisy signal. Although a great deal of smoothing takes place, the estimate that results is still a good representation of the original signal. The CV estimate in Figure 3(b), with $\hat{\alpha} = 1$ and $\hat{\lambda} = 150$, once again remains consistent with the noisy data. The error values in Table 1 once again show that the Bayesian estimate is a better approximation of the original signal.

If the parameter values selected by each method are examined closely, some generalizations can be made. The Bayesian parameter estimation technique tends to pick large values for both α and λ. Relatively small values of the Huber function threshold are selected if little smoothing is needed, along with correspondingly large values for λ. Conversely, larger values of α and smaller values of λ are selected to smooth the observed data more extensively. CV tends to always pick a very small value for α, along with a small value for λ. A small threshold value for α implies a consistent fit with the observed data, while a small value for λ implies a smooth signal estimate. These competing parameters result in estimates which closely follow the data, as evidenced in Figures 2 and 3. It can be inferred from the experimental results that CV fails to correctly estimate the threshold parameter.

Estimation Technique	σ	$\hat{\alpha}$	$\hat{\lambda}$	MSE	MAE
Bayesian $\hat{\mathbf{z}}$	10	82	1700	0.000277	0.009906
CV $\hat{\mathbf{z}}$	10	1	200	0.000644	0.012984
Bayesian $\hat{\mathbf{z}}$	20	103	800	0.000428	0.013262
CV $\hat{\mathbf{z}}$	20	1	150	0.000719	0.016082

Table 1: Comparison of parameter estimation techniques.

5 SUMMARY

An arbitrary choice can be made for selecting an objective function to use with the Gibbs density signal model. Each of the available objective functions contains a threshold, shape, or scaling parameter α which must be estimated, along with the temperature parameter λ of the density. Bayesian parameter estimation appears to be a feasible alternative to more classic techniques such as cross-validation in selecting stochastic signal model parameters. CV failed in selecting the correct parameter values for the estimation experiments in this paper. Several issues need to be addressed in continuing research. The parameter space Θ is currently optimized by global search; it would be more computationally appealing to incorporate a gradient-based method for obtaining the parameter estimate. The parameter prior model $\Pr(\theta|\mathcal{M})$ is currently a uniform density. If some prior information can be obtained regarding the parameters, a more informative Gaussian or skewed density might be used in its place. A related research topic lies in the area of model comparison [8, 9]; Bayesian techniques can also be used to select the best model for a given set of data, so that selecting a particular objective function and consistency measure becomes quantitative rather than subjective.

REFERENCES

[1] A. Blake and A. Zisserman, *Visual Reconstruction*, Cambridge, MA: The MIT Press, 1987.

[2] C. Bouman and K. Sauer, "An edge-preserving method for image reconstruction from integral projections," in *Proceedings of the Conference on Information Sciences and Systems*, pp. 382-387, The John Hopkins University, Baltimore, MD, March 20-22, 1991.

[3] D. Geman and G. Reynolds, "Constrained restoration and the recovery of discontinuities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 6, pp. 367-383, 1992.

[4] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, pp. 721-741, 1984.

[5] S. F. Gull, "Developments in maximum entropy data analysis," in *Maximum Entropy and Bayesian Methods*, J. Skilling, editor, Kluwer Academic Publishers, pp. 53-71, 1988.

[6] P. Hall and I. Koch, "On the feasibility of cross-validation in image analysis," *SIAM Journal on Applied Mathematics*, vol. 52, no. 1, pp. 292-313, 1992.

[7] S. Z. Li, "Reconstruction without discontinuities," in *Proceedings of the Third International Conference on Computer Vision*, pp. 309-312, Osaka, Japan, December, 1990.

[8] D. J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415-447, 1992.

[9] D. J. C. MacKay, "A practical bayesian framework for backprop networks," *Neural Computation*, vol. 4, no. 3, pp. 448-472, 1992.

[10] R. R. Schultz and R. L. Stevenson, "Improved definition image expansion," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pp. III-173 to III-176, San Francisco, CA, March, 1992.

[11] J. Skilling, "Classic maximum entropy," in *Maximum Entropy and Bayesian Methods*, J. Skilling, editor, Kluwer Academic Publishers, pp. 45-52, 1988.

[12] R. L. Stevenson and E. J. Delp, "Surface reconstruction with discontinuities," *Curves and Surfaces in Computer Vision and Graphics II*, in *Proceedings of the 1991 SPIE Conference*, pp. 46-57, Boston, MA, November 12-14, 1991.

[13] R. Szeliski, *Bayesian Modeling of Uncertainty in Low-level Vision*, Kluwer Academic Publishers, 1989.

[14] A. M. Thompson, J. C. Brown, J. W. Kay, and D. M. Titterington, "A study of methods of choosing the smoothing parameter in image restoration by regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 326-339, 1991.

[15] G. Wahba, "Spline bases, regularization, and generalized cross validation for solving approximation problems with large quantities of noisy data," in *Approximation Theory III: Proceedings of a Conference Honoring Professor George G. Lorentz*, E. W. Cheney, editor, New York: Academic Press, pp. 905-912, Austin, TX, January, 1980.

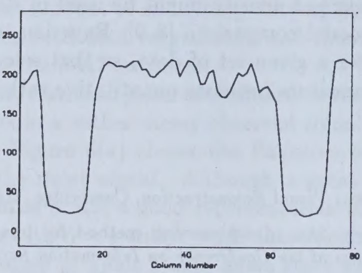
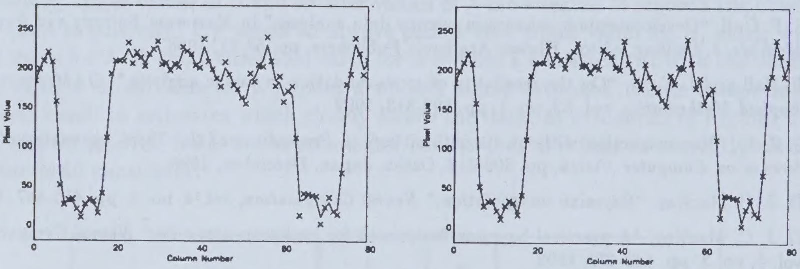
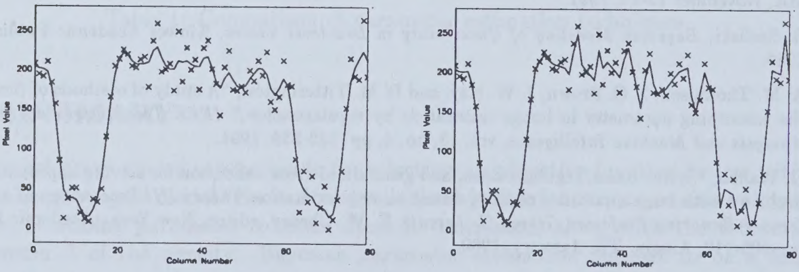


Figure 1: Original signal z.



(a) (b)

Figure 2: Estimates of noisy data ($\sigma = 10$); (a) Bayesian MAP estimate, $\hat{\alpha} = 82$, $\hat{\lambda} = 1700$; (b) CV estimate, $\hat{\alpha} = 1$, $\hat{\lambda} = 200$.



(a) (b)

Figure 3: Estimates of noisy data ($\sigma = 20$); (a) Bayesian MAP estimate, $\hat{\alpha} = 103$, $\hat{\lambda} = 800$; (b) CV estimate, $\hat{\alpha} = 1$, $\hat{\lambda} = 150$.

An Extension of Good's Roughness Penalty for Nonparametric Density Estimation[1]

Donald L. Snyder, Aaron Lanterman, and Michael I. Miller
Department of Electrical Engineering, Washington University, St. Louis, Missouri 63130

Summary

The unconstrained maximum-likelihood method for producing nonparametric estimates of density functions yields estimates that are unacceptably rough. While the expectation- maximization algorithm of Dempster, Laird, and Rubin [1] does permit maximum-likelihood estimates to be computed iteratively, the estimates produced at successive stages of the iteration display an increasing roughness and do not to converge towards any reasonable estimate as iterations proceed. The cause for this effect is that the unconstrained estimation-problem is fundamentally ill posed, requiring that some form of regularization be imposed to obtain acceptable solutions. When regularization is applied, the resulting density estimates no longer exhibit extreme roughness.

Good's roughness penalty [2] is used for regularizing nonparametric estimates of probability density functions [3] and radionuclide concentrations [4]-[6]. This penalty constrains the energy in the first derivative of the square root of the estimate, where the square root is used to enforce nonnegativity of the density estimate. In [7], we investigate an extension of Good's roughness penalty obtained by constraining energy in the n*th* derivative of the function. For estimating the joint probability-density of a pair of random variables or a two-dimensional radionuclide concentration, the extended penalty is equivalent to a Markov random field or Gibbs' prior in which the neighborhood of any pixel includes the 2n(n+1) closest pixels that surround it. Although the estimates that maximize likelihood penalized by extended Good's roughness are nonlinear, it is shown how n*th*-order Butterworth filters arise as natural smoothing kernels in one and two dimensions.

References

1. A. D. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *J. Royal Statistical Society*, Vol. B39, pp. 1-37, 1977.
2. I. J. Good and R. A. Gaskins, "Density Estimation and Bump-Hunting by the Penalized Likelihood Method Exemplified by Scattering and Meteorite Data," *J. American Statistical Association*, Vol. 75, pp. 42-73, March 1980.
3. R. A. Tapia and J. R. Thompson, *Nonparametric Probability Density Estimation*, Johns Hopkins University Press, Baltimore MD, 1978
4. M. I. Miller and D. L. Snyder, "The Role of Likelihood and Entropy in Incomplete-Data Problems: Applications to Estimating Point-Process Intensities and Toeplitz Constrained Covariances," *Proceedings of the IEEE*, Vol. 75, pp. 892-907, July 1987.
5. M. I. Miller and B. Roysam, "Bayesian Image Reconstruction for Emission Tomography Incorporating Good's Roughness Prior on Massively Parallel Processors," Proceedings of the National Academy of Sciences, April 1991.
6. D. L. Snyder and M. I. Miller, *Random Point Processes in Time and Space*, Ch. 3, Springer-Verlag, New York, 1991.
7. D. L. Snyder, A. Lanterman, and M. I. Miller, Regularizing Images in Emission Tomography via an Extension of Good's Roughness Penalty, *IEEE T. on Medical Imaging*, in review.

---

[1] This work was supported by the National Science Foundation under Grants Numbered MIP-9101991 and ECE-8552518 the National Institutes of Health under Grant RR01380 from the Division of Research Resources.

On the Asymptotic Stability of m-Switched Systems

Philippos Peleties and Raymond DeCarlo
School of Electrical Engineering,
Purdue University, W. Lafayette, IN 47907

Abstract

This paper is an investigation of asymptotic stability of *m*-switched systems based on Lyapunov functions. An *m*-switched system is a system $\dot{x} = A_{i_k}x$ where $A_{i_k} \in \{A_1, \ldots, A_m\}$ and i_k is an index from a switching sequence, i.e., control is achieved by switching between possible A_i-matrices. The association of a Lyapunov function with each A_i partitions the state-space into regions, called Ω-regions where the energy of $\dot{x} = A_i x$ decreases. Allowing for non-empty intersections between Ω-regions beyond the origin, and coverage of the state-space by their union enables us to consider switching within these intersections. The paper considers questions such as: existence of Ω-regions; inclusion of one Ω-region by another; coverage of the state-space by the union of these regions. Answering these questions is necessary in satisfying a theorem and its associated conditions for asymptotic stability of an *m*-switched system. A 2-switched system case study is finally explored.

1. Introduction

In [1] the investigation of the asymptotic stabilization of *m*-switched systems through Lyapunov functions was initiated. An *m*-switched system consists of *m* linear and autonomous systems

$$\dot{x}(t) = \sum_{i=1}^{m} \alpha_i(t) A_i x(t) \tag{1}$$

where $x \in \boldsymbol{R}^n$, $\alpha_i : \boldsymbol{R} \to \{0, 1\}$, is piecewise continuous, $\sum_{i=1}^{m} \alpha_i(t) = 1 \ \forall t$, where $\alpha_i(\tau_1) = 1$ implies that system A_i is actuated at time τ_1 with infinite switching being excluded. Each system A_i is in general unstable.

The dynamics of (1) are chosen by a switching sequence $s(x_0, \tau_0) = (i_0, \tau_0), (i_1, \tau_1), \ldots, (i_{k-1}, \tau_{k-1}), (i_k, \tau_k), \ldots$ where i_j indicates the actuation of system A_{i_j} with associated switching time τ_j, given the initial state x_0 and the initial time τ_0. More precisely, system A_i is actuated, at time τ_j if $\alpha_i(\tau_j^-) = 0$ and $\alpha_i(\tau_j) = 1$, i.e., the dynamics of (1) are governed at time τ_j by the system $\dot{x}(t) = A_i x(t)$. The objective is to asymptotically stabilize system (1).

2. Sufficient Conditions for Global Asymptotic Stability

Global asymptotic stability requires that the state of system (1) asymptotically approaches the origin $\forall x(\tau_0) \triangleq x_0$, i.e., $\lim_{t\to\infty} x(t) = 0$, through a series of switchings. Note that bold zero, $\mathbf{0}$, represents the origin of the state-space $\boldsymbol{R}^n$, and $\|x\|$ represents the Euclidean norm of vector x. Most of the proof of this papers can be found in [3] and [4].

Theorem 1: Consider system (1). Let $\pi_1(s(x_0, \tau_0)) = i_0, i_1, \ldots, i_{k-1}, i_k, \ldots$ identify the sequence of indices and $\pi_2(s(x_0, \tau_0)) = \tau_0, \tau_1, \ldots, \tau_{k-1}, \tau_k, \ldots$ identify the switching times associated with the switching sequence $s(x_0, \tau_0)$. Moreover, let $\{V_i(x) \mid i = 1, \ldots, m\}$, where $V_i(x)$ is associated with system A_i, be a set of functions which are real positive definite on $\boldsymbol{R}^n$ and radially unbounded. Suppose $\forall x_0 \in \boldsymbol{R}^n$ and all τ_0, there exists a switching sequence $s(x_0, \tau_0) = (i_0, \tau_0), (i_1, \tau_1), \ldots, (i_{k-1}, \tau_{k-1}), (i_k, \tau_k), \ldots$, and there exists constants $\rho > 0$, $\tau_{\min} > 0$ such that

(i) $\tau_{\min} \le \tau_j < \infty \ \forall \tau_j \in \pi_2(s(x_0, \tau_0))$,

(ii) $V_{i_{j+1}}(x(\tau_{j+2})) - V_{i_j}(x(\tau_{j+1})) \le -\rho \|x(\tau_{j+1})\|^2, \ \forall i_j, i_{j+1} \in \pi_1(s(x_0, \tau_0))$.

Then, the system (1) is globally asymptotically stable.

Since in general, each A_i is not a stability matrix, no radially unbounded and positive definite Lyapunov function exists having a negative definite time derivative along the trajectories of $\dot{x} = A_i x$ throughout the entire state-space. At best, there is a Lyapunov function, $V_i(x)$, whose time derivative is negative definite in the interior of some region, denoted by Ω_i, of the state-space. In such a region, the energy of the system, measured by its corresponding Lyapunov function $V_i(x)$, decreases. Theorem 1 says that if at switching points the energy of the system just prior to switching proportionately exceeds the energy of the system just prior to the next switching instant, then the system is asymptotically stabilizable by some switching sequence.

These Ω-regions are defined as follows:

Definition 1: Let a region $\Omega_i = \Omega_i^0 \cup \Omega_i^- \subset \boldsymbol{R}^n$ be such that $x \in \Omega_i^-$ (Ω_i^0) if and only if $\dot{V}_i(x) \triangleq \dfrac{\partial V_i}{\partial x} A_i x < 0$ $(= 0)$, $i = 1, \ldots, m$.

Theorem 1 places no requirements on Ω-regions, as defined by the previous definition. However, it is desirable that:

Condition 1: The state-space is "covered" by the totality of the Ω-regions, i.e., $\bigcup_{i=1}^{m} \Omega_i = \boldsymbol{R}^n$.

This condition parallels the idea of state-space coverage by the Ω-region corresponding to a stability matrix: if quadratic Lyapunov functions are considered, then there exists an Ω-region which covers the entire-state space.

For the purpose of automatic switching, it is advantageous to emulate a property of Variable Structure Systems (VSS) ([6, 7]), by requiring that switching occur when state trajectories cross manifolds contained within the overlap of "adjacent" Ω-regions. Hence, it is necessary that Ω-regions satisfy

Condition 2: Whenever $\Omega_i \cap \Omega_j - \{0\} \neq \emptyset$, then

(i) $\mathrm{int}(\Omega_i \cap \Omega_j) \neq \emptyset$, where "int" stands for *interior*, and

(ii) there exists an interior cone, $\mathbf{L}_{i,j} \subset \mathrm{int}(\Omega_i \cap \Omega_j) \cup \{0\}$ $(\{0\} \in L_{i,j})$, which does not contain any elements of either Ω_i^0 or Ω_j^0 with the exception of the origin, on whose boundary switching from system A_i to system A_j can occur.

Hence for the practical application of Theorem 1, Condition 1, and Condition 2 must be satisfied.

3. Properties of Ω-Regions

A number of elementary questions and properties are associated with each Ω-region:

1) Existence of Ω-regions:
   It can be shown that that the only matrix A_i for which one cannot construct an Ω-region by using a suitable r.s.p.d. matrix P_i is a positive scalar multiple of the identity. Note that the origin is always included in any possible Ω-region, therefore our contention here is for Ω-region containing more that the origin.

2) Geometric properties of Ω-regions:
   Given quadratic Lyapunov functions each Ω-region is a "cone" having "smooth" Ω^0, which most often corresponds to the boundary of the Ω-region. Moreover the intersection of Ω-regions is also a "cone". Additionally, if $\partial(\Omega_i) - \{0\} \neq \emptyset$ and $\mathrm{int}(\Omega_i) \neq \emptyset$, then $\partial(\Omega_i) = \Omega_i^0$ and $\mathrm{int}(\Omega_i) = \Omega_i^-$.

3) Relation of Ω-regions with eigenstructure of A matrices:
   The eigenvectors of a matrix A_i have a special relation with respect to Ω-regions associated with A_i and any r.s.p.d. P_i: the real eigenvectors associated with negative, zero, or positive eigenvalues of A_i belong to Ω_i^-, Ω_i^0, or $\boldsymbol{R}^n - \Omega_i$ respectively.

4) System trajectory characteristics with respect to Ω-regions:
   There are no equilibrium states in Ω_i^-. As a consequence, the trajectory $x_i(t_1; x_0, \tau_0) \triangleq e^{A_i(t_1 - \tau_0)} x_0$ will *always* approach Ω_i^0. Again we emphasize that generically Ω_i^0 corresponds to the boundary of Ω_i. Furthermore the trajectories of system A_i cannot escape to infinity within Ω_i^-.

4. Further Results on Ω-Regions

In our quest for finding conditions on A so that Theorem 1, Condition 1, and Condition 2 are satisfied, a number of more elaborate questions and properties surface, such as Ω-regions with non-empty interiors, inclusion of Ω-regions, and coverage of the state-space.

4.1. Existence of Ω-Regions with Non-empty Interiors

As previously noted, it is important to strengthen earlier results and investigate properties beyond those above. First there is a need to specify conditions for an Ω-region to have a non-empty interior. Mathematically, for an arbitrary $A \neq 0$ does there exist a r.s.p.d. P such that the associated Ω-region has a non-empty interior? The question here is: for an arbitrary A and a fixed $x \in \boldsymbol{R}^n$, does there exist a suitable r.s.p.d. P such that the associated Ω-region has a non-empty interior with x being a member of that interior? It turns out that as long as x is not an eigenvector associated with a non-negative real eigenvalue of A one can create a suitable P. Moreover, $x \in \Omega^-$.

Generically, the sets Ω^-, Ω^0, and $\boldsymbol{R}^n - \Omega$ correspond to the interior, boundary, and exterior of Ω from the topological point of view. However, there are some pathological cases of non-stability matrices for which the above is not true.

The problem arises from the fact that $Q \triangleq A^tP + PA$ is not indefinite: it is the existence of both positive and negative eigenvalues which creates boundaries in the usual topological sense. If $A^tP + PA$ is indefinite, then $\Omega^- = \mathrm{int}(\Omega)$, $\Omega^0 = \partial(\Omega)$, and $\boldsymbol{R}^n - \Omega \neq \emptyset$. It can be shown that as long as A is not a scalar multiple of the identity, a choice of r.s.p.d. P exists which can produce such an indefinite matrix Q:

Theorem 2: Let $A \in \boldsymbol{R}^{n \times n}$, $A \neq 0$. Suppose $x \in \boldsymbol{R}^n$ is such that $\nexists\, \alpha$ satisfying $Ax = \alpha x$. Then, $\exists\, P$, r.s.p.d., such that $\Omega^- \neq \emptyset$, $\Omega^0 - \{0\} \neq \emptyset$, and $\boldsymbol{R}^n - \Omega \neq \emptyset$.

Since one can always create a r.s.p.d. P so that its associated Ω-region has non-empty Ω^-, Ω^0, and $\boldsymbol{R}^n - \Omega$, we will henceforth only deal with this class of P's and consequent Ω-regions. Hence, $\Omega^- = \mathrm{int}(\Omega)$, and $\Omega^0 = \partial(\Omega)$.

4.2. Inclusion and Enlargement of Ω-Regions

The next task is that of finding Ω-regions which cover the state-space, as required by Condition 1. As a preliminary, it is natural to ask: *what is the "extent" or "size" of each Ω-region?* This question can be better understood when one already has a number of Ω-regions whose union does not cover the state-space: *how can one enlarge the regions so that coverage occurs?*

This implicitly requires the ability to compare two Ω-regions to decide which one is "bigger". What does "bigger" mean? Since the word "bigger" implies an ordering, the above question makes sense if one Ω-region "includes" or "contains" the other. Mathematically, the inclusion property implies that $\Omega_i \subset \Omega_j$ if and only if $\forall x \in \Omega_i$, then $x \in \Omega_j$. If expressed in terms of the signs of the functions x^tQ_ix and x^tQ_jx which define Ω_i and Ω_j respectively, then $\Omega_i \subset \Omega_j$ if and only if "$x^tQ_ix \leq 0$ implies that $x^tQ_jx \leq 0 \ \forall x \in \Omega_i$".

Determination of inclusion of Ω-regions based on the above description entails the computation of Ω_i^0 and Ω_j^0, a task that is not realistic for large-dimensioned Q. The following theorem answers the question of determining inclusion for two Ω-regions in more realistic terms. But first the following lemma is used in Theorem 3. This lemma asserts that if Ω_i is included within Ω_j, then the ratio $\mu(x) \triangleq \dfrac{x^tQ_ix}{x^tQ_jx}$ attains a maximum for some x inside Ω_i, i.e., $x \in \Omega_i^-$, and a minimum for some x outside Ω_j^-, i.e., $x \in \boldsymbol{R}^n - \Omega_j^-$.

Lemma 1: Let $P_i, P_j \in \boldsymbol{R}^{n \times n}$ be two r.s.p.s.d. or r.s.p.d. matrices such that $Q_i \triangleq A_i^tP_i + P_iA_i$ and $Q_j \triangleq A_j^tP_j + P_jA_j$ are indefinite. Assume that $\Omega_i \subset \Omega_j$ and $\Omega_i^0 \cap \Omega_j^0 - \{0\} = \emptyset$. Define $\mu(x) \triangleq \dfrac{x^tQ_ix}{x^tQ_jx}$, $\boldsymbol{S}_i = \{x \in \Omega_i \mid \|x\| = 1\}$, and $\boldsymbol{S}_j = \{x \in \Omega_j \mid \|x\| = 1\}$. Then

1. $\exists\, x_2 \in \boldsymbol{S}_i$ such that $\mu(x_2) \geq \mu(x) \ \forall x \in \boldsymbol{S}_i$. Moreover $\mu(x_2) > 0$.
2. $\exists\, x_1 \in \boldsymbol{B} - \boldsymbol{S}_j^-$ such that $\mu(x_1) \leq \mu(x) \ \forall x \in \boldsymbol{B} - \boldsymbol{S}_j^-$, where $\boldsymbol{B} \triangleq \{x \in \boldsymbol{R}^n \mid \|x\| = 1\}$, and $\boldsymbol{S}_j^- \triangleq \{x \in \Omega_j^- \mid \|x\| = 1\}$. Moreover $\mu(x_1) > 0$.

Strictly speaking the above minimum and maximum were found for $\boldsymbol{S}_i$ and $\boldsymbol{B} - \boldsymbol{S}_j^-$. However, note that $\mu(x_1) = \mu(\alpha_1 x_1)\ \forall \alpha \neq 0$ and $\mu(x_2) = \mu(\alpha x_2)\ \forall \alpha \neq 0$, making them minimum and maximum of Ω_i and $\boldsymbol{R}^n - \Omega_j^-$ respectively.

Theorem 3: Let $A_i, A_j \in \boldsymbol{R}^{n\times n}$ be two matrices and P_i, P_j, Q_i, Q_j, Ω_i, and Ω_j satisfy the conditions of Lemma 1. Then, $\Omega_i \subset \Omega_j$ if and only if $\exists\ \alpha \geq 0$ such that $Q_i - \alpha Q_j$ is r.s.p.s.d.

The above theorem required that $\Omega_i^0 \cap \Omega_j^0 - \{0\} = \emptyset$. We would like to relax this condition and consider the general case. The following proposition does so.

Proposition 1: Theorem 3 is valid even if $\Omega_i^0 \cap \Omega_j^0 - \{0\} \neq \emptyset$.

The following algorithm checks the applicability of Theorem 3 on two Ω-regions. The algorithm establishes the existence (or lack) of parameter α of the theorem in question, and it is based on generalized eigenvalues with respect to the pair (Q_i, Q_j) [8].

Algorithm 1:

Step 1. Compute the generalized eigenvalues with respect to the pair (Q_i, Q_j), i.e., let α_k, $k = 1, \ldots, n$ indicate these generalized eigenvalues, i.e., $\operatorname{rank}[Q_i - \alpha_k Q_j] < n$.

Step 2. If $\nexists$ real $\alpha_k \geq 0$, then stop. Theorem 3 is not satisfied; otherwise for some real $\alpha_k \geq 0$, compute the eigenvalues of $Q_i - \alpha_k Q_j$. Let λ_r, $r = 1, \ldots, n$ indicate these eigenvalues.

Step 3. Theorem 3 is satisfied if and only if $\lambda_r \geq 0$, $\forall r = 1, \ldots, n$; if this condition fails return to Step 2 and choose another real $\alpha_k \geq 0$, if possible; otherwise stop.

The above theorem and its associated algorithm provide us with a way of computing inclusion of two known Ω-regions. It is now important to discuss extensions of an Ω-region to arrive at a "bigger" one. The constructive proof of the following proposition, found in [3] provides a process for increasing the "extent" of a given Ω-region.

Proposition 2: Let $A_i \in \boldsymbol{R}^{n\times n}$ be a non-stability matrix and consider a r.s.p.d. matrix $P_i \in \boldsymbol{R}^{n\times n}$ such that $Q_i \triangleq A_i^t P_i + P_i A_i$ is not a definite matrix. Then, $\exists\ \hat{P}_i$, r.s.p.s.d, such that $\Omega_i \subset \hat{\Omega}_i$, $\Omega_i \neq \hat{\Omega}_i$ where $\hat{Q}_i \triangleq A_i^t \hat{P}_i + \hat{P}_i A_i$.

4.3. Coverage of State-Space by Ω-Regions

Following the "inclusion" question is the "coverage" question as posed at the beginning of this section. The obvious conditions for the coverage of the state-space by a union of Ω-regions is given by the following proposition.

Proposition 3: Let $Q_k \in \boldsymbol{R}^{n\times n}$, $k = 1, \ldots, m$ and Ω_k be its corresponding Ω-region. We say that $\bigcup_{k=1}^{m} \Omega_k = \boldsymbol{R}^n$ if and only if $\forall x \in \boldsymbol{R}^n$ there exists at least on index j, which is x-dependent, such that $x^t Q_j x \leq 0$.

The following is a direct result for two Ω-regions.

Corollary 1: Let $Q_i, Q_j \in \boldsymbol{R}^{n\times n}$ and Ω_i, Ω_j their corresponding Ω-regions. We say that $\Omega_i \cup \Omega_j = \boldsymbol{R}^n$ if and only if $x^t Q_i x > 0$, implies $x^t Q_j x \leq 0$.

As with Theorem 3, a realistic method for checking for coverage of the state-space is needed. Theorem 4 does so and it is based on Theorem 3. First, the following two lemmas are used in Theorem 4.

Lemma 2: Let Q_i, Q_j be two indefinite matrices. If $\Omega_i = \Omega_j$, then $\exists\ \alpha > 0$ such that $Q_i = \alpha Q_j$.

Lemma 3: Let Q_k be indefinite matrices associated with Ω_k, $k = i, j$. If $\Omega_i \cup \Omega_j = \boldsymbol{R}^n$, then either $\Omega_i^0 = \Omega_j^0$ or $\Omega_i^0 \cap \Omega_j^0 - \{0\} = \emptyset$.

The obvious generalization of the above lemma for more than two Ω-regions does not necessarily hold.

Theorem 4: Let $A_i, A_j \in \boldsymbol{R}^{n\times n}$ be two matrices and P_i, P_j be r.s.p.s.d. or r.s.p.d. such that Q_i, Q_j are indefinite. Then, $\Omega_i \cup \Omega_j = \boldsymbol{R}^n$ if and only if $\exists\ \alpha \geq 0$ such that $Q_i + \alpha Q_j$ is r.s.n.s.d.

As with Theorem 3 and Algorithm 1, the following algorithm allows one to check the applicability of Theorem 4 on two Ω-regions. The algorithm establishes the existence (or lack) of the α of the theorem in question and again is based on generalized eigenvalues with respect to pairs $(Q_i, -Q_j)$.

Algorithm 2:

Step 1. Compute the generalized eigenvalues with respect to the pair $(Q_i, -Q_j)$, i.e., let α_k, $k = 1, \ldots, n$ indicate these generalized eigenvalues, i.e., $\text{rank}[Q_i - \alpha_k Q_j] < n$.

Step 2. If $\nexists$ real $\alpha_k \geq 0$, then stop. Theorem 4 is not satisfied; otherwise for some $\alpha_k \geq 0$ compute the eigenvalues of $Q_i + \alpha_k Q_j$. Let λ_r, $r = 1, \ldots, n$ indicate those eigenvalues.

Step 3. Theorem 4 is satisfied if and only if $\lambda_r \leq 0$, $\forall r = 1, \ldots, n$; if this condition fails return to Step 2 and choose another $\alpha_k \geq 0$, if possible.

5. Case Study: 2-Switched System

When there is switching between two systems, then one can obtain stronger results than previously attained. The assumption here then is that there are only two systems having matrices $A_1, A_2 \in \boldsymbol{R}^{n \times n}$ with both A_1 and A_2 unstable, neither of which is a scalar multiple of the identity.

Variable Structure Systems (VSS) ([6, 7]) provide a natural framework for attempting to asymptotically stabilize such a system. The idea here is to locate an $n-1$-dimensional hyperspace which is globally attractive *and* the "average" system restricted on the hyperspace is stable. Once located, definite quadratic Lyapunov functions are created having Ω-regions whose intersection properly contains the hyperspace, hence guaranteeing the existence of an "envelope" around the hyperspace for the switched system to evolve in a stable fashion. Care is taken for the Ω-regions to satisfy Theorem 1, Condition 1, and Condition 2.

In answering the question of the existence of globally attractive $n-1$-dimensional hyperspace, definite quadratic Lyapunov functions are considered. Having said that, we first seek necessary and sufficient condition for the global attractivity to an $n-1$-dimensional hyperspace. However, it is customary in VSS to use semi-definite quadratic Lyapunov functions measuring system energy relative to the hyperspace. Extension from semi-definite to definite quadratic Lyapunov functions will be accomplished later in the paper. The necessary and sufficient conditions for global asymptotic attractivity are provided by the following proposition. As noted it utilizes well known notions of VSS. All the proofs of this section can be found in [4].

Proposition 4: Consider $A_1, A_2 \in \boldsymbol{R}^{n \times n}$,
$s \in \boldsymbol{R}^{1 \times n}$, $s \neq 0$. The $n-1$-dimensional hyperspace $sx = 0$ is globally asymptotically attractive if and only if $\exists\, \alpha < 0$, $\beta > 0$ such that $\alpha s = sA_1 + \beta s A_2$.

The question of computing such an $n-1$-dimensional hyperspace, given the two matrices A_1 and A_2 of this problem, comes up. However, since it is also required that the "average" system--a suitable convex combination of A_1 and A_2--restricted on the hyperspace is stable, the search for this hyperspace is modified so as to satisfy *both* global asymptotic attractivity *and* stability. This is accomplished by Algorithm 4. This algorithm is based on computing the crossings of the imaginary axis by the eigenvalues of some linear positive combination of matrices A_1 and A_2. Let us elaborate on this. The eigenvalues of the linear combination $A_1 + \alpha A_2$ vary as a function of α. Consider the real parts of these eigenvalues. For some value of α, at least one of these eigenvalues either crosses or "touches" the imaginary axis, i.e., its real part becomes zero. Consider the collection of all such αs. Denote this collection with Γ. Observe that Γ has a finite number of elements. Hence, by computing Γ, we can determine, in a finite number of steps, all possible sign changes of the real part of these eigenvalues. Therefore, the question of whether there exists a suitable α such that $A_1 + \alpha A_2$ has eigenvalues with a particular mix of positive, negative, and zero real parts can be answered in a finite number of steps. Algorithm 4 computes Γ.

The set Γ includes, among others, the generalized eigenvalues with respect to the pair $(A_1, -A_2)$[8]: these are points where the real eigenvalues of the linear combination become zero, one eigenvalue per generalized eigenvalue.

The remaining elements of Γ are, of course, associated with complex conjugate pairs of the eigenvalues of $A_1 + \alpha A_2$: the real part of the complex conjugate pair becomes zero. Algorithm 3 is used in determining such elements. Since Algorithm 4 computes the entire set Γ, among others, and Algorithm 3 is used to compute part of Γ, it is presented first. Moreover, it is presented in its most general form. Briefly, it computes the purely imaginary (if they exist) eigenvalues of $A_1 - c_1 A_2$, where c_1 is real.

Algorithm 3 [5]:

Step 1. Let $R_1 \triangleq I_2 \otimes A_2$, $R_2 \triangleq \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \otimes I_n$, $[\, c_0^1 \;\; c_0^2 \,] = [\, 0 \;\; 0 \,]$, and $M_0 \triangleq I_2 \otimes A_1 - \hat{c}^1 R_1 - \hat{c}^2 R_2$, where $[\, \hat{c}^1 \;\; \hat{c}^2 \,]$ is an initial guess of the solutions, and $\otimes$ denotes Kronecker product. This is the initialization part and $k = 0$ the first time in Step 2.

Step 2. Compute the SVD of M_k, $M_k = U_k S_k V_k^t$. If $\| c_k^1 R_1 + c_k^2 R_2 \|_2 > \varepsilon \| M_k \|_F$, where F stands for Frobenious norm, then stop. The "best" solution is $[\, c_k^1 + \hat{c}^1 \;\; c_k^2 + \hat{c}^2 \,]$; otherwise continue to Step 3. Note that the solution is such that it minimizes $\sigma_{\min}(M_k)$, the smallest singular value of M_k. If $\sigma_{\min}(M_k) = 0$, then the solution is a true one, otherwise it is a "minimizing" one in the sense explained.

Step 3. Compute $[\delta c_k^1 \;\; \delta c_k^2] = \sigma_{\min}(M_k) \begin{bmatrix} u_{k,\min}^t R_1 v_{k,\min} \\ u_{k,\min}^t R_2 v_{k,\min} \end{bmatrix}^+$, where $u_{k,\min}$, $v_{k,\min}$ are the left and right singular vectors corresponding to $\sigma_{\min}(M_k)$, and + represents the Moore-Penrose pseudo inverse.

Step 4. Set $[c_{k+1}^1 \;\; c_{k+1}^2] = [c_k^1 \;\; c_k^2] + [\delta c_k^1 \;\; \delta c_k^2]$.

Step 5. Set $M_{k+1} = M_0 - c_{k+1}^1 R_1 - c_{k+1}^2 R_2$. Go to Step 2.

Given A_1, A_2 and Algorithm 3, the following algorithm computes α, β, and s of Proposition 4 with the additional result of the "average" system, restricted on the hyperspace $sx = 0$, being asymptotically stable.

Algorithm 4:

Step 1. Compute the generalized eigenvalues with respect to the pair $(A_1^t, -A_2^t)$. Let $\sigma(A_1^t, -A_2^t)$ be the set of these generalized eigenvalues.

Step 2. Compute all $c^1 \triangleq c_k^1 + \hat{c}^1$ by using Algorithm 3. Use $\hat{c}^1 \in \sigma(A_1^t, -A_2^t)$ as initial guesses. Let $\boldsymbol{C}$ be the set of all $-c^1$.

Step 3. Let $\Gamma = \text{ordered } (\sigma(A_1^t, -A_2^t) \cup \boldsymbol{C})$ and let δ be a fixed positive number such that $\delta \leq \min(\gamma_{i+1} - \gamma_i)$, $\gamma_i, \gamma_{i+1} \in \Gamma$. Define $\boldsymbol{B} \triangleq \{\beta_i \mid \beta_1 = \gamma_1 - \delta,\ \beta_2 = \gamma_1 + \delta,\ \ldots,\ \beta_{i+1} = \gamma_i + \delta,\ i = 2, \ldots, |\Gamma| \}$.

Step 4. If $\forall\, \beta_i \in \boldsymbol{B}$, real, $\beta_i \leq 0$, then stop. There is no globally asymptotically attractive hyperspace $sx = 0$; otherwise for some real $\beta_i \in \boldsymbol{B}$, $\beta_i > 0$, compute the eigenvalues of $A_1^t + \beta_i A_2^t$. Let $\sigma(A_1^t + \beta_i A_2^t)$ be the set of those eigenvalues.

Step 5. If $\exists\, \alpha \in \sigma(A_1^t + \beta_i A_2^t)$, real, $\alpha > 0$, then even if there exists a globally asymptotically attractive hyperspace $sx = 0$ the system $A_1 + \beta_i A_2$ when restricted to the hyperspace will not be asymptotically stable. Go to Step 4 and choose the next real element $\beta_{i+1} \in \boldsymbol{B}$; otherwise for some $\alpha < 0$ compute the associated eigenvector s^t. Define $\beta \triangleq \beta_i$. Then, $(A_1^t + \beta A_2^t) s^t = \alpha s^t$ satisfies Proposition, and the "average" system $A_1 + \beta A_2$ is asymptotically stable when restricted on the hyperspace $sx = 0$.

Define $\alpha_0 \triangleq 1/(1 + \beta)$, where β is as per Step 4 of Algorithm 4, and multiply the "average" system $A_1 + \beta A_2$ by α_0. Clearly, the combination $A_{eq}(\alpha_0) \triangleq \alpha_0 A_1 + (1 - \alpha_0) A_2$ is a convex one in tradition with VSS, where $A_{eq}(\alpha_0)$ is the scaled "average" system. Moreover, the $n-1$-dimensional hyperspace $sx = 0$ is $A_{eq}(\alpha_0)$-invariant and $A_{eq}(\alpha_0)$ when restricted to $sx = 0$ is asymptotically stable. From this point on, $A_{eq}(\alpha_0)$ will represent the "average" system.

Clearly, since the above two conditions, the first of invariance of the $n-1$-dimensional hyperspace and the other the stability of the "average" system restricted on this hyperspace, are satisfied, a r.s.p.d. matrix P_{eq} exists such that the projection of $Q_{eq}(\alpha_0) \triangleq A_{eq}^t(\alpha_0) P_{eq} + P_{eq} A_{eq}(\alpha_0)$ onto $sx = 0$ is r.s.n.d. Naturally, in satisfying Theorem 1, Condition 1, and Condition 2, it is desirable that the hyperspace $sx = 0$ lie in the interior of $\Omega_1 \cap \Omega_2$, where Ω_i is associated with A_i. This requires that there exist r.s.p.d. P_i, $i = 1, 2$ such that $Q_i \triangleq A_i^t P_i + P_i A_i$ is r.s.n.d. when projected onto the hyperspace $sx = 0$. Could it be possible to progressively modify P_{eq} so that P_i is produced with the desired properties? What are the conditions involving $A_{eq}(\alpha_0)$, A_1, and A_2 so that at least the "local" version of Theorem 1 is satisfied? By "local" we mean "in the vicinity of the $n-1$-dimensional hyperspace $sx = 0$". Such conditions which involve the above quantities are offered by Proposition 5. But first, the following lemma is used in Proposition 5.

Lemma 4: If $R^t[A_{eq}^t(\alpha_0)P_i + P_iA_{eq}(\alpha_0)]R$ is r.s.n.d., then $R^t(e^{A_{eq}^t(\alpha_0)}P_ie^{A_{eq}(\alpha_0)} - P_i)R$ is r.s.n.d.

Proposition 5: Let $R^t(A_i^tP_i + P_iA_i)R$ be r.s.n.d. $i = 1, 2$. If $R^t[A_{eq}^t(\alpha_0)P_i + P_iA_{eq}(\alpha_0)]R$ is r.s.n.d., then a "local" version of Theorem 1 is being satisfied, in the vicinity of the $n-1$-dimensional hyperspace $sx = 0$.

Note that if $sx = 0$ is globally asymptotically attractive, then Theorem 1 is satisfied in its global form. Such is the case here since the hyperspace was computed through Algorithm 4, guaranteeing global asymptotic attractivity.

The previous discussion has presented a number of ideas which are useful in attempting to device a suitable algorithm for constructing P_1 and P_2 which satisfy Theorem 1, Condition 1, and Condition 2. Such is Algorithm 5. It is based on the idea of iteratively modifying a r.s.p.d. P_{eq} whose Ω-region corresponding to $A_{eq}(\alpha_0)$ properly contains the hyperspace $sx = 0$. This modification allows one to arrive at a new r.s.p.d. matrix P_i, whose two Ω-regions, one corresponding to A_i and the other corresponding to $A_{eq}(\alpha_0)$, also contain the hyperspace. The algorithm is run twice, once for A_1 and again for A_2.

During the actual construction of P_i, the parameter α of $A_{eq}(\alpha)$ is varied. In essence, one starts with P_{eq} and $A_{eq}(\alpha)|_{\alpha=\alpha_0}$ and increases (if dealing with A_1) or decreases (if dealing with A_2) α so that the Ω-region of P_{eq} corresponding to $A_{eq}(\alpha)$ has a boundary point on the hyperspace. This occurs because the eigenvalues of $Q_{eq}(\alpha)$ change as a function of α in a continuous fashion. At one point, the parameter α is such that one of the eigenvalues of the projection of $Q_{eq}(\alpha)$ onto the hyperspace becomes zero. This implies that the Ω-region defined by $Q_{eq}(\alpha)$ indeed has a boundary point on the hyperspace. Then, appropriate perturbations are added to P_{eq} resulting in a new P. These perturbations not only preserve the positive definiteness of P, but also restore the property that the hyperspace is properly contained in the intersection of the two new Ω-regions, one corresponding to P and $A_{eq}(\alpha_0)$ and the other corresponding to P and $A_{eq}(\alpha)$. This occurs because the additive perturbations on P_{eq} push the zero eigenvalue of the projection of $Q_{eq}(\alpha)$ onto the hyperspace further inside the left half plane.

The algorithm terminates when α has reached one (zero) in which case the Ω-regions of the modified P_{eq} corresponding to $A_{eq}(\alpha_0)$ and $A_{eq}(1) = A_1$ ($A_{eq}(\alpha_0)$ and $A_{eq}(0) = A_2$) properly contain the hyperspace. As such, the provisions of Proposition are satisfied. Any computational aspects of the algorithm such as initialization and determination of various parameters will be discussed in what follows after the algorithm. But first the following definition.

Definition 2: Let P be r.s.p.s.d., and Q be r.s.n.s.d. Define

$$\beta^+(P, -\Delta P) \triangleq \begin{cases} +\infty & \text{if } \Delta P \text{ is r.s.p.s.d. or r.s.p.d.} \\ \lambda_{\min}^+(P, -\Delta P) & \text{otherwise} \end{cases}$$

and

$$\beta^+(Q, -\Delta Q) \triangleq \begin{cases} +\infty & \text{if } \Delta Q \text{ is r.s.n.s.d. or r.s.n.d.} \\ \lambda_{\min}^+(Q, -\Delta Q) & \text{otherwise} \end{cases}$$

where $\lambda_{\min}^+(A, B)$ is the minimum positive generalized eigenvalue with respect to the pair (A, B).

Algorithm 5:

Step 1. Let $\tilde{P}_k$ be r.s.p.d. and α_k be such that $Q_{k,k} \triangleq R^t[A_{eq}^t(\alpha_k)\tilde{P}_k + \tilde{P}_kA_{eq}(\alpha_k)]R$, and $Q_{k,0} \triangleq R^t[A_{eq}^t(\alpha_0)\tilde{P}_k + \tilde{P}_kA_{eq}(\alpha_0)]R$ are r.s.n.s.d. and r.s.n.d. respectively, the first having one eigenvalue equal to zero. If $\alpha_k > 1$ ($\alpha_k < 0$), then stop and set $P_1 = \tilde{P}_k$ ($P_2 = \tilde{P}_k$), otherwise, continue to Step 2.

Step 2. Let $\{\hat{P}_i, i = 1, \ldots, \frac{n(n+1)}{2}\}$ be a set of r.s. basis matrices spanning the space of n-dimensional r.s. matrices, and let $\hat{Q}_{i,k} \triangleq R^t[A_{eq}^t(\alpha_k)\hat{P}_i + \hat{P}_iA_{eq}(\alpha_k)]R$, and $\hat{Q}_{i,0} \triangleq R^t[A_{eq}^t(\alpha_0)\hat{P}_i + \hat{P}_iA_{eq}(\alpha_0)]R$. In addition, let $(\lambda_{\min}(\tilde{P}_k), e_p(\tilde{P}_k))$, $(\lambda_{\max}(Q_{k,k}), e_q(Q_{k,k}))$, and $(\lambda_{\max}(Q_{k,0}), e_q(Q_{k,0}))$, be eigenvalue/eigenvector pairs where $\lambda_{\min}(\tilde{P}_k) = \min_{i=1..n}\lambda_i(\tilde{P}_k)$, $\lambda_{\max}(Q_{k,k}) = \max_{i=1..n-1}\lambda_i(Q_{k,k}) = 0$, and $\lambda_{\max}(Q_{k,0}) = \max_{i=1..n-1}\lambda_i(Q_{k,0})$.

Step 3. Form the matrix $M_k \triangleq [m_{ij}]$, $i = 1, \ldots, 3$, $j = 1, \ldots, \frac{n(n+1)}{2}$ where $m_{1j} = e_p^t(\tilde{P}_k)\hat{P}_je_p(\tilde{P}_k)$,

$m_{2j} = e_q^t(Q_{k,k})\hat{Q}_{j,k}e_q(Q_{k,k})$, and $m_{3j} = e_q^t(Q_{k,0})\hat{Q}_{j,0}e_q(Q_{k,0})$. In addition, let $u \triangleq [u_i]$, $i = 1, \ldots, 3$ be such that $u_1 = 1$, $u_2 = -1$, and $u_3 = -1$. Compute $\gamma_k = M_k^+ u$, where M_k^+ is the Moore-Penrose pseudo inverse of M_k.

Step 4. Form $\Delta\tilde{P}_k = \sum_{i=1}^{\frac{n(n+1)}{2}} \gamma_{k,i}\hat{P}_i$, $\Delta Q_{k,k} = \sum_{i=1}^{\frac{n(n+1)}{2}} \gamma_{k,i}\hat{Q}_{i,k}$, and $\Delta Q_{k,0} = \sum_{i=1}^{\frac{n(n+1)}{2}} \gamma_{k,i}\hat{Q}_{i,0}$.

Step 5. Let $D(\tilde{P}_k) = \{\delta \mid \lambda_{\min}(\tilde{P}_k + \delta\Delta\tilde{P}_k) = \max_{0 \le l \le \beta^+(\tilde{P}_k, -\Delta\tilde{P}_k)} \lambda_{\min}(\tilde{P}_k + l\Delta\tilde{P}_k)\}$, $D(Q_{k,k}) = \{\delta \mid \lambda_{\max}(Q_{k,k} + \delta\Delta Q_{k,k}) = \min_{0 \le l \le \beta^+(Q_{k,k}, -\Delta Q_{k,k})} \lambda_{\max}(Q_{k,k} + l\Delta Q_{k,k})\}$, and $D(Q_{k,0}) = \{\delta \mid \lambda_{\max}(Q_{k,0} + \delta\Delta Q_{k,c}) = \min_{0 \le l \le \beta^+(Q_{k,0}, -\Delta Q_{k,0})} \lambda_{\max}(Q_{k,0} + l\Delta Q_{k,0})\}$. Set $\delta_k = \min(\max(D(\tilde{P}_k)) \cup \max(D(Q_{k,k})) \cup \max(D(Q_{k,0})))$.

If $\delta_k = \infty$, then set $\tilde{P}_k = \Delta\tilde{P}_k$, $Q_{k,k} = \Delta Q_{k,k}$, $Q_{k,0} = \Delta Q_{k,0}$, and $\delta_k = 0$.

Step 6. Set $\tilde{P}_{k+1} = \tilde{P}_k + \delta_k\Delta\tilde{P}_k$, $Q_{k+1,k} = Q_{k,k} + \delta_k\Delta Q_{k,k}$, and $Q_{k+1,0} = Q_{k,0} + \delta_k\Delta Q_{k,0}$.

Step 7. Let $\alpha_{k+1} > \alpha_k$ $(\alpha_{k+1} < \alpha_k)$ be such that $Q_{k+1,k+1}$ is r.s.n.s.d. If no such α_{k+1} exists, then set $\alpha_{k+1} > 1$ $(\alpha_{k+1} < 0)$. Go to Step 1.

Initializing the above algorithm entails the computation of a r.s.p.d. $\tilde{P}_0 \triangleq P_{eq}$ such that $Q_{0,0} = R^t\left[A_{eq}^t(\alpha_0)P_{eq} + P_{eq}A_{eq}(\alpha_0)\right]R$, the projection of $Q_{eq}(\alpha_0)$ onto the hyperspace, is r.s.n.s.d. However, we can do better than that. Since the hyperspace $sx = 0$ is $A_{eq}(\alpha_0)$-invariant, and $A_{eq}(\alpha_0)$ is globally asymptotically stable when restricted onto the hyperspace, we can compute a P_{eq} such that $Q_{0,0}$ is r.s.n.d. The following method does so.

Initialization of Algorithm 5:

Step 1. Compute $M_{eq}(\alpha_0) \in \boldsymbol{R}^{n-1 \times n-1}$ such that $A_{eq}(\alpha_0)R = RM_{eq}(\alpha_0)$. One such $M_{eq}(\alpha_0)$ can be computed as $M_{eq}(\alpha_0) = \left[(A_{eq}(\alpha_0)R)^+R\right]^{-1}$, where + stands for the Moore-Penrose pseudo inverse. Note that $\sigma(M_{eq}(\alpha_0)) \subset C^-$ since the hyperspace $sx = 0$ is $A_{eq}(\alpha_0)$-invariant and $\sigma(A_{eq}(\alpha_0))\,|_{sx=0} \subset C^-$.

Step 2. Solve the Lyapunov equation $M_{eq}(\alpha_0)^t P + PM_{eq}(\alpha_0) = Q$ for $P \in \boldsymbol{R}^{n-1 \times n-1}$ where $Q \in \boldsymbol{R}^{n-1 \times n-1}$ is some r.s.n.d. It is preferable but not required that $Q = -I_{n-1}$.

Step 3. Compute $P_{eq} \triangleq (R^+)^t PR^+ + s^t s$ where R^+ is the Moore-Penrose pseudo inverse of R.

Since $Q_{0,0}$ is r.s.n.d., the first iteration of the algorithm can bypass Steps 2 through 6 and go directly to Step 7. Here, we set $Q_{1,0} = Q_{0,0}$ and $\tilde{P}_1 = \tilde{P}_0$ as to keep the subscripts consistent with being at the 0th iteration through Algorithm 5. Write $Q_{1,0} = R^t[A_2^t\tilde{P}_1 + \tilde{P}_1 A_2]R + \alpha_0 R^t[(A_1 - A_2)^t\tilde{P}_1 + \tilde{P}_1(A_2 - A_1)]R \triangleq B_1 + \alpha_0 B_2$ and apply the following simple algorithm:

Implementation of Step 7 of Algorithm 5:

Step 1. Compute the generalized eigenvalues with respect to the pair $(B_1, -B_2)$. Let β_m, $m = 1, \ldots, n$ denote these generalized eigenvalues.

Step 2. Let $\beta_r > \alpha_k$ and $\beta_r \le \beta_m \ \forall \beta_m > \alpha_k$ if we are computing P_1, or $\beta_r < \alpha_k$ and $\beta_r \ge \beta_m \ \forall \beta_m < \alpha_k$ if we are computing P_2. Set $\alpha_{k+1} = \beta_r$.

Observe that the generalized eigenvalues with respect to a symmetric pair are real whenever they exist. Furthermore, as α is varied beyond α_k the largest (least negative) eigenvalue of $B_1 + \alpha B_2$ as α will cross the zero axis at that generalized eigenvalue which is closest to α_k.

Subsequent iterations entail the optimizations of Steps 5. These optimizations can be easily carried out through the bisection method. The following algorithm applies the bisection method to the first optimization of Step 5. The other two optimizations are carried out in a similar way.

Implementation of Step 5 of Algorithm 5 (bisection method):

Step 1. If $\beta^+(P_k, -\Delta P_k) = +\infty$, then set $\delta = +\infty$ and stop; otherwise set $b_{l,0} = \lambda_{\min}(P_k)$, $b_{u,0} = \lambda_{\max}(P_k) + \beta^+(P_k, -\Delta P_k)\lambda_{\max}(\Delta P_k)$, $\omega_0 = \tfrac{1}{2}(b_{l,0} + b_{u,0})$.

Step 2. Compute the generalized eigenvalues with respect to the pair $(P_k - \omega_i I, -\Delta P_k)$. Let $B_i = \{\beta_1, \ldots, \beta_n\}$ denote the set of these generalized eigenvalues.

Step 3. If $\exists\, \beta_r \in B_i$, real, such that $\lambda_{\min}(P_k + \beta_r \Delta P_k) = \omega_i$, then set $b_{l,i+1} = \omega_i$, $b_{u,i+1} = b_{u,i}$, $\omega_{i+1} = \frac{1}{2}(b_{l,i+1} + b_{u,i+1})$; otherwise if for every real $\beta \in B_i$, $\lambda_{\min}(P_k + \beta \Delta P_k) < \omega_i$ or B_i contains no real β, set $b_{l,i+1} = b_{l,i}$, $b_{u,i+1} = \omega_i$, $\omega_{i+1} = \frac{1}{2}(b_{l,i+1} + b_{u,i+1})$.

Step 4. If $|b_{u,i+1} - b_{l,i+1}| > \varepsilon$ go to Step 2; otherwise continue.

Step 5. Set $\delta = \beta_r$, $\lambda_{\min}(P_k + \delta P_k) = \omega_{i+1}$.

Clearly, the successful computation of P_i, $i = 1, 2$ through Algorithm 5 guarantees that Theorem 1 is satisfied, as shown by Proposition 5. Moreover

1. $\Omega_i^- \neq \emptyset$ since $x^t Q_i x = x^t(A_i^t P_i + P_i A_i)x < 0$, $i = 1, 2$ for $x \neq 0$, $sx = 0$.

2. $\Omega_1 \cap \Omega_2 - \{0\} \neq \emptyset$ since the above x satisfies $x \in \Omega_i^-$, $i = 1, 2$. Therefore,

- **(i)** $\operatorname{int}(\Omega_1 \cap \Omega_2) \neq \emptyset$, hence satisfying Condition 1, and
- **(ii)** from the continuity of quadratic Lyapunov functions and their time derivative on x, $\exists\, \boldsymbol{L}_{1,2} \triangleq \{x \in \operatorname{int}(\Omega_1 \cap \Omega_2)\}$ such that switching can occur on the boundary of $\boldsymbol{L}_{1,2}$, $\partial \boldsymbol{L}_{1,2}$, hence satisfying Condition 2.

The computation of P_1 and P_2 through Algorithm only guarantees local asymptotic stability inside the envelope created by $\Omega_1 \cap \Omega_2$. It does not guarantee that $\Omega_1 \cup \Omega_2 = \boldsymbol{R}^n$. However global state-space coverage is required by Condition 1. Let us remind ourselves that using $x^t s^t s x$ as a Lyapunov function guarantees the global asymptotic attractivity of the $n-1$-dimensional hyperspace $sx = 0$ as per Proposition. Can these two ideas, computation of P_i guaranteeing local asymptotic stability and $s^t s$ guaranteeing global asymptotic attractiveness of the hyperspace $sx = 0$, be combined so that one can arrive at two new r.s.p.d. matrices $\bar{P}_1$ and $\bar{P}_2$ which satisfy the same conditions as P_1 and P_2 *and* their associated Ω-regions satisfy $\bar{\Omega}_1 \cup \bar{\Omega}_2 = \boldsymbol{R}^n$? In other words, can these new $\bar{P}_1$ and $\bar{P}_2$ guarantee global asymptotic stability? The answer is yes. The following proposition shows why.

Proposition 6: Given the P_i computed by Algorithm 5, $\exists\, \gamma > 0$ such that Theorem 1, Condition 1, and Condition 2 are satisfied for $\bar{P}_i \triangleq s^t s + \gamma P_i$.

The following algorithm computes a γ for Proposition 6.

Algorithm 6:

Step 1. Compute $\alpha \geq 0$ such that $\tilde{Q}_1 + \alpha \tilde{Q}_2$ is r.s.n.s.d. where $\tilde{Q}_i = A_i^t s^t s + s^t s A_i$. Such an α can be computed through Algorithm 4.

Step 2. Apply Algorithm 4 to compute $\tilde{\gamma} > 0$ such that $\tilde{Q}_1 + \alpha \tilde{Q}_2 + \tilde{\gamma}(Q_1 + \alpha Q_2)$ is r.s.n.s.d., where $Q_i = A_i^t P_i + P_i A_i$. Choose γ such that $0 < \gamma < \tilde{\gamma}$.

In [4] an example of a six-dimensional 2-system is presented. Unfortunately we are not able to include it here due to space limitations.

6. Summary

This paper has investigated some aspects of the asymptotic stabilization of m-switched systems, made of m linear, autonomous systems, through Lyapunov functions. A switching sequence describes the structure alteration of the switched system, and for each switching time it specifies the new dynamics. One Lyapunov function is associated with each one of the m systems. A sufficient condition on the difference of the Lyapunov functions between successive switchings guaranteeing asymptotic stability of the switched system is that it is negatively upper bounded. The association of Lyapunov functions with the m systems yields a partitioning of the state-space into energy-decreasing Ω-regions. Non-zero intersections of these regions and coverage of the state-space by their union are desirable conditions for automatic switching. This switching occurs within these non-zero intersections.

A number of interesting properties are associated with Ω-regions: (i) conic structure, (ii) set-theoretic inclusion and exclusion of the eigenvectors of each corresponding system matrix associated with negative and positive eigenvalues respectively, and (iii) exclusion of all equilibrium points of each associated system from the interior of its corresponding Ω-region.

Moreover, additional questions that were considered were: (i) existence of Ω-regions, (ii) inclusion of Ω-regions, (iii) coverage of the state-space by their union, and (iv) structural conditions on the time derivatives of the Lyapunov functions Q, so that state-space coverage is accomplished. The emphasis when answering these questions was on testable conditions. It has been shown that: regarding existence of Ω-regions, (i) they always exist as long as a system matrix is not a positive scalar multiple of the identity, and (ii) these Ω-regions have an interiors, boundaries, and exteriors in the usual topological sense; regarding Ω-region inclusion, (i) one Ω-region is included within another if and only if a proper linear combination of their Qs is r.s.p.s.d, (ii) testability is facilitated through generalized eigenvalues, and (iii) one can "enlarge" an Ω-region by additive perturbations on P; regarding state-space coverage (i) the union of two Ω-regions covers the state-space if and only if a proper linear combination of their Qs is r.s.n.s.d, (ii) testability is facilitated through generalized eigenvalues, and (iii) extension of (i) to m Ω-regions is applicable only in its sufficiency. Finally, the necessary structural condition on two Qs to cover the state-space is that the sum of the number of their non-positive eigenvalues exceeds the dimension of the state-space.

The paper concluded with the investigation ofa 2-switched system case study. The effort in this study was to emulate desirable properties from Variable Structure Systems: a globally asymptotically attractive hyperspace was designed where the "average" system-- a suitable convex combination of the two system matrices A_1 and A_2--when restricted on this hyperspace, stayed there for all subsequent time and was asymptotically stable. Then, two r.s.p.d. matrices, P_1 and P_2 were computed so that the hyperspace was properly contained in the intersection of their associated Ω-regions. Switching occurred inside this intersection at the boundaries of a properly contained cone.

References

[1] P. A. Peleties and R. A. DeCarlo, ''Asymptotic Stability of m-Switched Systems using Lyapunov-like Functions'', *The American Conference on Control*, Boston, June 1991.

[2] P. A. Peleties and R. A. DeCarlo, ''Asymptotic Stability of m-Switched Systems using Lyapunov Functions'', submitted to *IEEE Conference on Decision and Control.*

[3] P. A. Peleties and R. A. DeCarlo, ''Asymptotic Stability of m-Switched Systems using Lyapunov Functions: General Theory'', submitted to *Mathematics of Control, Signals, and Systems.*

[4] P. A. Peleties and R. A. DeCarlo, ''Asymptotic Stability of 2-Switched Systems using Lyapunov Functions: A Case Study,'' submitted for publication to the *Mathematics of Control, Signals, Systems.*

[5] M. A. Wicks and R. A. DeCarlo, ''Structured Rank-Reducing Matrix Perturbations: Theory and Computation,'' *Proceedings of the Twenty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September 1991.

[6] R. A. DeCarlo, S. H. Żak and G. P. Matthews, ''Variable Structure Control of Nonlinear Multivariable Systems : A Tutorial,'' *Proceedings of the IEEE*, Vol. 76, No. 3, March 1988, pp. 212-232.

[7] V. I. Utkin, ''Variable Structure Systems with Sliding Modes'', *IEEE Transactions on Automatic Control*, AC-22, No. 2, 1977, pp. 212-222.

[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, The John Hopkins University Press, 1989.

Robust Stability of Singularly Perturbed Systems

Z.Shao and M.E.Sawan
Department of Electrical Engineering
The Wichita State University
Wichita, KS 67208

Abstract: *In this paper, we will discuss the robust stability problem of linear time-invariant singularly perturbed systems, where the system matrix may have parameter perturbation bounded by H_∞-norm. A sufficient condition for the upper bound of ε is given, such that the stability of the actual system (or the full-order model) can be inferred from the analysis of the reduced order system (or the slow model) in separate time scales.*

I. Introduction and Problem Statement

Singularly perturbed systems, or two-time scale systems, often occur because of the presence of small parasitic parameters multiplying the time derivatives of some of the system states. As stability is a crucial system requirement, the stability problem (or ε-bound promblem) in singularly perturbed systems has been drawn much attention in the past. Consider the linear time-invariant singularly perturbed system given by

$$\dot{x}(t)=A_1x(t) + A_2z(t) + B_1u(t) \tag{1a}$$

$$\varepsilon\dot{z}(t)=A_3x(t) + A_4z(t) + B_2u(t) \tag{1b}$$

$$y(t)=C_1x(t) + C_2z(t) \tag{1c}$$

where $x \in \Re^n$, $z \in \Re^m$ are the state vectors of the slow and fast modes individully, $u \in \Re^l$ is the control vector, and $y \in \Re^p$ is the output vector. $A_i(i=1,2,3,4)$, B_j and $C_j(j=1,2)$ are appropriate dimension matrices, and the singular perturbation parameter ε is non-negative, and represents the response of the fast modes.

Set $\varepsilon=0$ in (1b), and substitute (1b) into (1a), we get a reduced-order model (or slow modes) with fast modes eliminated:

$$\dot{x}_s(t)=A_0x_s(t) + B_0u_s(t) \qquad (2)$$

where

$$A_0=A_1 - A_2A_4^{-1}A_3 \qquad (3a)$$

$$B_0=B_1 - A_2A_4^{-1}B_2 \qquad (3b)$$

then the stability problem of singularly perturbed system (1) is formulated as follow: if the reduced-order system (2) is stable, does it guarantee that 1) is also stable? This problem is of great practical significance for analysis and design because the stability of system (1) with (n+m)th-order can be infered from the analysis of system (2) with only nth-order in separate time scales. Since (2) is only an appropriate model of (1), the stable reduced-order system (2) does not imply the stability of the original full-order system (1), but, the following lemma gives the conditions under which the singularly perturbed system (1) is also asymptotically stable for small enough ε.

Lemma 1 [1]:

If both A_4 and A_0 are Hurwitz, then there exists an ε_0 such that for all $\varepsilon \in [0,\varepsilon_0)$ system (1) is asymptotically stable. ■

How to find the largest positive ε_0 such that , for all $\varepsilon \in [0,\varepsilon_0)$, system (1) is stable, and hence (2) is really a valid approximation of (1), has been given much attention, many upper bounds ε_0 have been calculted, via various ways, for the acceptable size of ε, see [1]-[5]. But, up to now, all the papers on the stability problem in singularly perturbed systems, discuss only the "nominal" model, that is, the system matrix is completely known. Actually, in real control systems, it is inevitable that the system matrix will have perturbation, due to component value variation, and so on. In this paper, we will discuss the robust stability problem of linear time-invariant singularly perturbed systems, where the system matrix may have parameter perturbation. That is, we will consider the following singularly perturbed system:

$$\dot{x}(t)=(A_1+\Delta_1)x(t) + (A_2+\Delta_2)z(t) + B_1u(t) \qquad (4a)$$

$$\varepsilon\dot{z}(t)=(A_3+\Delta_3)x(t) + (A_4+\Delta_4)z(t) + B_2u(t) \qquad (4b)$$

$$y(t)=C_1x(t) + C_2z(t) \qquad (4c)$$

where $\Delta_i(i=1,2,3,4)$ is appropriate dimension parameter perturbation corresponding to $A_i(i=1,2,3,4)$, and satisfies certain conditions. Then, the same stability problem as in system (1) is, how to find ε_0 as large as possible, such that if the reduced-order model of system (4) given by

$$\dot{x}_{s\Delta}(t)=A_{0\Delta}x_{s\Delta}(t) + B_{0\Delta}u_{s\Delta}(t) \tag{5}$$

is stable, also implies that system (4) is stable, where

$$A_{0\Delta}=(A_1+\Delta_1)-(A_2+\Delta_2)(A_4+\Delta_4)^{-1}(A_3+\Delta_3)$$

$$B_{0\Delta}=B_1-(A_2+\Delta_2)(A_4+\Delta_4)^{-1}B_2$$

or

$$A_{0\Delta}= A_0 + \varphi \tag{6a}$$

where

$$\varphi=\Delta_1-\Delta_2(A_4+\Delta_4)^{-1}(A_3+\Delta_3)-A_2(A_4+\Delta_4)^{-1}\Delta_3+A_2(A_4+\Delta_4)^{-1}\Delta_4A_4^{-1}A_3 \tag{6b}$$

II. Main Result

In this section, a sufficient condition is given for the acceptable size of ε, such that the stability of (4) can be infered from the reduced-order system (5). First, we present some lemmas which may be used in the proof of the main result.

Lemma 2[7]:

Let P and P+QRS be non-singular matrices, then

$$(P+QRS)^{-1} \equiv P^{-1} -P^{-1}Q(I+RSP^{-1}Q)^{-1}RSP^{-1} \qquad \blacksquare$$

Lemma 3.

If $M \in \Re^{n*n}$ is an Hurwitz matrix, then

i) $\|M^{-1}\|_\infty \le \|(sI_n-M)^{-1}\|_\infty < +\infty$;

ii) For $\varepsilon>0$, $\|A(sI_n-M)^{-1}B\|_\infty \equiv \|A(\varepsilon sI_n-M)^{-1}B\|_\infty$, where A and B are arbitrarily constant matrices;

iii) $\|s(sI_n-M)^{-1}\|_\infty < +\infty$.

where the H_∞-norm of a stable matrix G(s) is defined by

$$\|G(s)\|_\infty=\sup_\omega \sigma_{\max}[G(j\omega)]$$

$\sigma_{\max}$ is the maximum singular value of $G(j\omega)$.

Proof:

i) Because M is Hurwitz, so $(sI-M)^{-1}$ is asymptotically stable, it guarantees that $\|(sI-M)^{-1}\|_\infty$ is bounded.

Note that $\|M^{-1}\|_\infty=\|(j\omega I-M)^{-1}{}_{|\omega=0}\|_\infty$, so we have

$$\|M^{-1}\|_\infty\le\|(j\omega I-M)^{-1}\|_\infty$$

ii) Suppose $\|A(j\omega I-M)^{-1}B\|_\infty=\sigma_{\max}[A(j\omega I-M)^{-1}B]_{|\omega=\omega 0}$, then

$$\|A(j\varepsilon\omega I-M)^{-1}B\|_\infty = \sigma_{\max}[A(j\varepsilon\omega I-M)^{-1}B]_{|\omega=\omega 0/\varepsilon} \equiv \|A(j\omega I-M)^{-1}B\|_\infty$$

iii) Suppose $(sI-M)^{-1} = (a_{ij}(s))_{n*n}$, then $a_{ij}(s)$ takes the form as

$$a_{ij}(s) = (c_{ij}^{(1)}s^{n-1}+ \cdots +c_{ij}^{(n)})/(s^n+b_1 s^{n-1}+ \cdots +b_n)$$

where $b_l(l=1,2, \cdots ,n)$ and $c_{ij}^{(k)}(k=1,2, \cdots ,n)$ are appropriate constants, and $a_{ij}(s)$ is asymptotically stable. Then

$$s(sI-M)^{-1}=(sa_{ij}(s))_{n*n}, \quad \text{where}$$

$$sa_{ij}(s)=(c_{ij}^{(1)}s^n+\cdots+c_{ij}^{(n)}s)/(s^n+b_1 s^{n-1}+\cdots+b_n)$$

so $\|j\omega(j\omega I-M)^{-1}\|_\infty$ is bounded. ■

The following is the main result of this paper.

Theorem 1.

Suppose A_0 and A_4 are both Hurwitz in (1) and (2), assume that A_2 and A_3 are non-zero matrices, and the system matrix perturbation satisfies the following conditions i)-iv):

i) $\|\Delta_1\|_\infty \le h_1 \|A_2\|_\infty \|A_3\|_\infty \|A_4^{-1}\|_\infty$;

ii) $\|\Delta_i\|_\infty \le h_i \|A_i\|_\infty$ $(i=2,3)$;

iii) $\|\Delta_4\|_\infty \le h_4 \|(sI_m-A_4)^{-1}\|_\infty^{-1}$;

iv) $h_1 + (h_2+h_3+h_4+h_2h_3)/(1-h_4) \le h(\|A_2\|_\infty \|A_3\|_\infty \|A_4^{-1}\|_\infty \|(sI_n-A_0)^{-1}\|_\infty)^{-1}$

where $h_i(i=1,2,3,4)$ are nonnegative constants and $h_4<1$, $h<1$. Then, if

$$\varepsilon < c(1-h)(1-h_4)^2/[(1+h_2)(1+h_3)] \tag{7a}$$

system (4) is stable, where

$$c=(\|A_2\|_\infty \|A_3\|_\infty \|A_4^{-1}\|_\infty \|(sI_m-A_4)^{-1}\|_\infty \|s(sI_n-A_0)^{-1}\|_\infty)^{-1} \tag{7b}$$

Proof:

Taking Laplace transform of (4a) and (4b), we have

$$sX(s)=(A_1+\Delta_1)X(s)+x(0)+(A_2+\Delta_2)Z(s)+B_1U(s) \tag{8a}$$

$$Z(s)=(\varepsilon sI_m-A_4-\Delta_4)^{-1}(A_3+\Delta_3)X(s)+(\varepsilon sI_m-A_4-\Delta_4)^{-1}\varepsilon z(0)$$
$$+(\varepsilon sI_m-A_4-\Delta_4)^{-1}B_2U(s) \tag{8b}$$

where $x(0)$ and $z(0)$ are the initial states of $x(t)$ and $z(t)$, respectively. Substitute (8b) into (8a), and rearange (8a), we have

$$X(s)=T(s)x(0) + T(s)(A_2+\Delta_2)(\varepsilon sI_m-A_4-\Delta_4)^{-1}\varepsilon z(0)$$
$$+ T(s)[B_1+(A_2+\Delta_2)(\varepsilon sI_m-A_4-\Delta_4)^{-1}B_2]U(s) \tag{8c}$$

where

$$T(s)=[sI_n-A_1-\Delta_1-(A_2+\Delta_2)(\varepsilon sI_m-A_4-\Delta_4)^{-1}(A_3+\Delta_3)]^{-1} \tag{8d}$$

Use lemma 2 in (8d), and substitute (8d) into (8c), it can be shown that

$$X(s)=\Delta(s)(sI_n-A_0-\varphi)^{-1}x(0)+\Delta(s)(sI_n-A_0-\varphi)^{-1}(A_2+\Delta_2)(\varepsilon sI_m-A_4-\Delta_4)^{-1}\varepsilon z(0)$$

$$+ \Delta(s)(sI_n-A_o-\varphi)^{-1}[B_1+(A_2+\Delta_2)(\varepsilon sI_m-A_4-\Delta_4)^{-1}B_2]U(s) \tag{9a}$$

where

$$\Delta(s)=(I_n-\psi(s))^{-1} \tag{9b}$$

and

$$\psi(s)=s(sI_n-A_o-\varphi)^{-1}(A_2+\Delta_2)(A_4+\Delta_4)^{-1}[sI_m-(A_4+\Delta_4)/\varepsilon]^{-1}(A_3+\Delta_3) \tag{9c}$$

φ is the same as defined in (6c).

Next we show that

(a) $A_4+\Delta_4$ is Hurwitz, or $(\varepsilon sI_m-A_4-\Delta_4)^{-1}$ is asymptotically stable;

(b) $A_o+\varphi$ is Hurwitz, or $(sI_n-A_o-\varphi)^{-1}$ is asymptotically stable;

(c) $\|\psi(s)\|_\infty<1$;

(d) $\Delta(s)$ is asymptotically stable.

First we prove (a).

Because $(sI_m-A_4-\Delta_4)^{-1}\equiv(I_m-(sI_m-A_4)^{-1}\Delta_4)^{-1}(sI_m-A_4)^{-1}$ from i) of Lemma 3, we have

$$\|(sI_m-A_4)^{-1}\Delta_4\|_\infty \le \|(sI_m-A_4)^{-1}\|_\infty\|\Delta_4\|_\infty \le h_4 <1$$

Using the "Small gain theorem", this means $(I_m-(sI_m-A_4)^{-1}\Delta_4)^{-1}$ is asymptotically stable, then $(sI_m-A_4-\Delta_4)^{-1}$ is asymptotically stable, or $A_4+\Delta_4$ is Hurwitz.

(b) Noting that $(sI_n-A_o-\varphi)^{-1}\equiv(I_n-(sI_n-A_o)^{-1}\varphi)^{-1}(sI_n-A_o)^{-1}$ and it can be shown that $\|\varphi\|_\infty \le h\|(sI_n-A_o)^{-1}\|_\infty^{-1}$, so

$$\|(sI_n-A_o)^{-1}\varphi\|_\infty \le \|(sI_n-A_o)^{-1}\|_\infty\|\varphi\|_\infty \le h < 1$$

Still using the "Small gain theorem", we know $(I_n-(sI_n-A_o)^{-1}\varphi)^{-1}$ is asymptotically stable, then $(sI_n-A_o-\varphi)^{-1}$ is asymptotically stable, or $A_o+\varphi$ is Hurwitz.

(c) Noting that

$$\psi(s)\equiv s(sI_n-A_o)^{-1}X(A_2+\Delta_2)(A_4+\Delta_4)^{-1}(sI_m-A_4/\varepsilon)^{-1}Y(A_3+\Delta_3) \tag{10a}$$

where

$$X=(I_n-\varphi(sI_n-A_o)^{-1})^{-1} \tag{10b}$$

and

$$Y=(I_m-\Delta_4(\varepsilon sI_m-A_4)^{-1})^{-1} \tag{10c}$$

Because $X\equiv(sI_n-A_o)(sI_n-A_o-\varphi)^{-1}$, from (a) we know that X is stable, also $X-I\equiv\varphi(sI_n-A_o)^{-1}X$, so $\|X-I\|_\infty\le\|\varphi\|_\infty\|(sI_n-A_o)^{-1}\|_\infty\|X\|_\infty$.

Because $\|\varphi\|_\infty\le h\|(sI_n-A_o)^{-1}\|_\infty^{-1}$, so

$$\|X\|_\infty\le 1/(1-h)$$

(11a)

Using the similar method and ii) of lemma 3, we can prove that Y is stable and

$$\|Y\|_\infty\le 1/(1-h_4) \tag{11b}$$

Also

$$\|(A_4+\Delta_4)^{-1}-A_4^{-1}\|_\infty=\|(A_4+\Delta_4)^{-1}\Delta_4A_4^{-1}\|_\infty$$
$$\le h_4\|(A_4+\Delta_4)^{-1}\|_\infty\|A_4^{-1}\|_\infty\|(sI_m-A_4)^{-1}\|_\infty^{-1}$$

then

$$\|(A_4+\Delta_4)^{-1}\|_\infty\le\|A_4^{-1}\|_\infty/[1-h_4\|A_4^{-1}\|_\infty\|(sI_m-A_4)^{-1}\|_\infty^{-1}]$$
$$\le\|A_4^{-1}\|_\infty/(1-h_4) \qquad (11c)$$

Take H_∞-norm of both sides of (10a), substitute (11a),(11b),(11c) into (10a), and use the given condition, we have

$$\|\psi(j\omega)\|_\infty\le b\varepsilon\|A_2\|_\infty\|A_3\|_\infty\|A_4^{-1}\|_\infty\|(sI_m-A_4)^{-1}\|_\infty\|s(sI_n-A_o)^{-1}\|_\infty<1$$

where $b=(1+h_2)(1+h_3)/(1-h)(1-h_4)^2$.

(d) Using the "Small gain theorem", it is directly from iii). ■

Remarks:

i) If $\Delta_i=0$ $(i=1,2,3,4)$, we can set $h_i(i=1,2,3,4)=0$ and $h=0$, then if $\varepsilon < c$ the system (1) is asymptotically stable.

ii) If $A_2=0$ or $A_3=0$, we can use the similar method in Theorem 1 to get the corresponding upper bound for ε, but, some conditions may be changed, such as $\|\Delta_i\|_\infty\le h_i$ $(i=2,3)$ instead of condition ii). In this case, the form of the bounded condition for h_i $(i=1,2,3,4)$ and h is similar to, but not as fine as that of iv) in theorem 1.

iii) Theorem 1 also gives a sufficient condition to find the range of h_i and h, that is, if it is required that system (1) is stable for $\varepsilon\in[0,\varepsilon_o)$ where $\varepsilon_o<c$, then if h_i and h are chosen such that conditions i)-iv) of theorem 1 are satisfied, and that,

$$(1-h)(1-h_4)^2/[(1+h_2)(1+h_3)] > \varepsilon_o/c \qquad (12)$$

system (4) will be also stable. We call this problem as "inverse" problem, because we are from $\varepsilon < \varepsilon_o < c$ to find acceptable upper bound of the system matrix perturbation.

III. Examples

Example 1. For singularly perturbed system

$$\begin{vmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\\ \varepsilon\dot{z}_1(t)\\ \varepsilon\dot{z}_2(t)\end{vmatrix}=\begin{vmatrix}-1/2 & 0 & -1 & 0\\ 0 & -1/2 & 0 & -1\\ 1 & 0 & -3 & a\\ 0 & 1 & b & -3\end{vmatrix}\begin{vmatrix}x_1(t)\\ x_2(t)\\ z_1(t)\\ z_2(t)\end{vmatrix}$$

where a, b are constants.

i) Prove if $\varepsilon\in[0,4)$ the system is stable when $a=b=0$;

ii) If $\varepsilon\in[0,2)$, then $\max\{|a|,|b|\}\le1.0$ will guarantee that the system is still stable.

Proof:

i) It is easy to find that

$$\|A_2\|_\infty=\|A_3\|_\infty=\|(sI_n-A_0)^{-1}\|_\infty=\|s(sI-A_0)^{-1}\|_\infty=1$$

$$\|A_4^{-1}\|_\infty=\|(sI-A_4)^{-1}\|_\infty=0.5$$

from (7b) we know c=4, or we can say, if $\varepsilon\epsilon[0,4)$, the system is asymptotically stable.

ii) From the conditions in Theorem 1 and (12), set $h_1=h_2=h_3=0$, we know if

$$0\le h_4, h<1 \tag{a}$$

$$h_4/(1-h_4)<2h \tag{b}$$

$$(1-h)(1-h_4)^2>2/c \tag{c}$$

the system is asymptotically stable.

The above sufficient conditions (a)-(c) are equivalent to the following conditions (d)-(f):

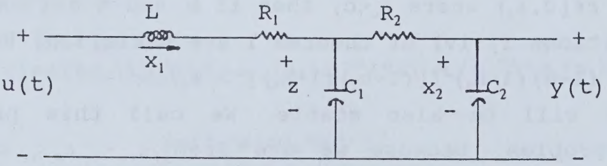
$$h_4<1-1/\sqrt{2(1-h)} \tag{d}$$

$$0\le h_4, h<1 \tag{e}$$

$$h_4<2h/(1+2h) \tag{f}$$

Let $1-1/\sqrt{2(1-h)}=2h/(1+2h)$, we find $h=0.151$, then $h_4\le0.532$, from iii) of theorem 1 we know, if $\|\Delta_4\|_\infty\le h_4\|(sI-A_4)^{-1}\|_\infty^{-1}$, or $\max\{|a|,|b|\}\le1.0$, the system is asymptotically stable. ■

Example 2. Consider the following *R-L-C* network



where L=0.1 H, C_2=0.1 F, $R_1\le$0.05 Ω, R_2=1 Ω ± 5% due to the environmenttemperature. C_1 is a very small parameter. Use the result of theorem 1 to find the range for C_1, such that the system is stable.

Solution:

Suppose the state variables are as shown in the above figure, and $C_1=\varepsilon$, then it is easy to derive the state equation as follow:

$$\begin{vmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \varepsilon\dot{z}(t) \end{vmatrix} = \begin{vmatrix} -R_1/L & 0 & -1/L \\ 0 & -1/R_2C_2 & 1/R_2C_2 \\ 1 & 1/R_2 & -1/R_2 \end{vmatrix} \begin{vmatrix} x_1(t) \\ x_2(t) \\ z(t) \end{vmatrix} + \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} u(t) \qquad (13a)$$

If we set $R_1 = 0$, and $R_2 = 1$, we get the following nominal system:

$$\begin{vmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \varepsilon\dot{z}(t) \end{vmatrix} = \begin{vmatrix} 0 & 0 & -10 \\ 0 & -10 & 10 \\ 1 & 1 & -1 \end{vmatrix} \begin{vmatrix} x_1(t) \\ x_2(t) \\ z(t) \end{vmatrix} + \begin{vmatrix} 1 \\ 0 \\ 0 \end{vmatrix} u(t) \qquad (13b)$$

and $\|A_1\|_\infty = 10$, $\|A_2\|_\infty = 14.14$, $\|A_3\|_\infty = 1.414$, $\|A_4^{-1}\|_\infty = 1$

$\|(sI-A_4)^{-1}\|_\infty = 1$, $\|(sI-A_0)^{-1}\|_\infty \leq 0.26226$, $\|s(sI-A_0)^{-1}\|_\infty \leq 1.5$

from (8) we find

$$c = 1/30 \qquad (14)$$

It can be shown that the perturbations Δ_i $(i=1,2,3,4)$ defined in (4a) and (4b) satisfy the following inequalities:

$\|\Delta_1\|_\infty \leq 10/19$; $\|\Delta_2\|_\infty \leq 10/19$; $\|\Delta_3\|_\infty \leq 1/19$; $\|\Delta_4\|_\infty \leq 1/19$

then we find $h_1 = 1/38$; $h_2 = h_3 = 0.037$; $h_4 = 1/19$; $h = 0.8492$

from (7a) and (14) we know, if $C_1 \leq 4.35*10^{-3}$ F, the system is stable.

IV. Conclusion

An explicit upper bound for ε is given to ensure the robust stability of a linear time-invariant singularly perturbed system, in the sense the system matrix may have parameter perturbation. Since the result is based on the nominal system matrix in appropriate parameter perturbation, it also applies to completely known systems. This paper also offers a way to find the range for system matrix perturbation if the stability of the system is required for $\varepsilon \epsilon [0, \varepsilon_0)$, where $\varepsilon_0 < c$. This result may be very useful in system design and analysis, if we know the range of ε in advance. A problem of interest is to investigate how to use composite feedback control from slow and fast models to stabilize system (4) with certain parameter perturbation. Work in this direction is in progress.

References

[1] P.V.Kokotovic, H.K.Khalil, and J.O'Reilly, *Singular perturbation in control: analysis and methods*. NY: Academic,1986

[2] N.R. Sandell, Jr. "Robust stability of systems with

application to singular perturbation", *Automatica*, Vol.15, pp.467-470, 1979

[3] Bor-Sen Chen and Chun-Liang Lin, "On the stability bounds of singularly perturbed systems", *IEEE Automat. Contr*. No.11, Nov. pp.1265-1270, 1990

[4] W.Feng, "Characterization and computation for the bound ε in linear time-invariant singularly perturbed systems", *Syst. Contr. Letl.*, Vol.11, pp.195-202, 1988

[5] V.R.Saksena, J.O'Reilly, and P.V.Kokotovic, "Singular perturbations and time-scale methods in control theory: Survey 1976-1983". *Automatica*, Vol.20, pp.273-293, 1984

[6] M. Vidyasagar, *Control system synthesis: A Factorization Approach*. Cambridge, MA: M.I.T press, 1985

[7] T.C.Hsia, *System identification*. Boston, MA, Lexington, 1977

[8] R.Lane Dailey,"*Lecture notes for the workshop on robust control*". *ACC*, May 1990, San Diego, CA

[9] J.M.Maciejouski, *Multivariable Feedback Design*. Addision-wesley publishing Company, 1989

Exponential Stabilization of Uncertain Systems by Dynamic Feedback

Engin Yaz & Asad Azemi
Electrical Engineering Department
University of Arkansas
Fayetteville, AR 72701

Abstract. A full-order dynamic compensation scheme is presented for stabilization of a class of uncertain nonlinear systems. The gains used in the nonlinear compensator are like those in variable structure designs but continuous in nature. The stability of the controlled system is theoretically proved and illustrated by an example.

1. Introduction

The control of systems with imperfectly known dynamics continues to be an important problem for control theorists. Such inaccurate models are largely motivated by numerous diverse application possibilities. One of the outstanding design methods for these systems is the variable structure or min-max controllers. Some important references in this respect include [1]-[3] on state feedback designs, [4]-[6] on static output feedback designs. For a system whose state vector is not completely available for measurement, one obvious choice is the use of observer-based control policies. References [7]-[9] are on variable structure observer design and [10]-[13] are concerned with issues arising in observer-based variable structure controllers.

In this work, we propose a novel nonlinear observer-based controller (compensator) for exponential stabilization of nonlinear uncertain systems. First, the observer will be introduced and then the control design will be based on that. Next, a generalization of the observer to the case where there are measurement uncertainties/nonlinearities will be given. Finally, the theoretical results obtained are illustrated with an example. We remark here that, even though the nonlinear gains used in this work resemble those of min-max observers and controller ones, they will be continuous in nature permitting us to avoid the analysis of differential equations with discontinuous right hand sides, and achieving exponential bounding of the estimation error and state trajectories by simple Lyapunov type arguments.

In this paper $\mathbf{R}_+$ is the set of nonnegative real numbers and $\mathbf{R}^n$ denotes the n-dimensional real Euclidean space with norm $\|x\| \doteq (x^Tx)^{1/2}$. $\|A\|_s$ is the induced matrix norm for $A \in \mathbf{R}^{n\times n}$, where $\|A\|_s = \overline{\lambda}^{1/2}(A^TA)$, $\overline{\lambda}(\underline{\lambda})$ denoting the maximum (minimum) eigenvalue of the symmetric matrix. $A>0$, I, $A^{\#}$ respectively denote a positive definite matrix (in the sense of Loewner ordering), an identity matrix whose dimension is determined from its context, and the Moore-Penrose pseudo inverse of rectangular matrix A.

2. Preliminary Results

In this section, we present three lemmas which will later be used in proving the main results.

Lemma 1. Suppose that we are given $A \in \mathbf{R}^{n\times n}$, $B \in \mathbf{R}^{n\times m}$, $Q \in \mathbf{R}^{n\times m}$, $Q=Q^T>0$ and $P \in \mathbf{R}^{n\times n}$, $P=P^T>0$ that is to assigned as a solution to the algebraic Lyapunov equation

$$(A+BK)^TP+P(A+BK) = -Q$$

This work was done when the first author was on leave at Space Systems Control Lab, Purdue University, West Lafayette, IN 47907 by some choice of the gain matrix $K \in \mathbf{R}^{m\times n}$. P is assignable as a solution if an only if P satisfies

$$(I - BB^{\#})(P^{-1}A^T + AP^{-1} + P^{-1}QP^{-1})(I - BB^{\#}) = 0$$

in which case, the gain K that achieves this assignment is not unique and is given by

$$K = -B^{\#}(P^{-1}A^T + AP^{-1} + P^{-1}QP^{-1})\left(I - \frac{1}{2}BB^{\#}\right)P - B^{\#}SP + (I - B^{\#}B)Z$$

where S is a skew-symmetric matrix that satisfies $(I - BB^{\#})S = 0$ and Z is arbitrary.
Proof. The proof can be made by simple modification of the one in [14].

Similar to the above, one can prove the following lemma :

Lemma 2. For the assignability of the solution $P = P^T > 0$ to

$$(A - GC)^T P + P(A - GC) = -Q$$

with $C \in \mathbf{R}^{p\times n}$, by a matrix $G \in \mathbf{R}^{n\times p}$, it is necessary and sufficient that P satisfy

$$(I - C^{\#}C)(A^T P + PA + Q)(I - C^{\#}C) = 0$$

in which case the assignment can be realized by

$$G = P^{-1}\left[\left(I - \frac{1}{2}C^{\#}C\right)(A^T P + PA + Q) + S\right]C^{\#} + Z(I - C^{\#}C)$$

where again S is a skew symmetric matrix and satisfying $S(I - C^{\#}C) = 0$ and Z is arbitrary.

Lemma 3. Consider the nonlinear system

$$\dot{x} = f(x\,,t) \qquad , \qquad x(t_0) = x_0$$

where $x \in \mathbf{R}^n$. Suppose there exists a matrix $P = P^T > 0$ for this system such that

$$\frac{d}{dt}[x^T(t)Px(t)] \le -\gamma_1\|x(t)\|^2 + \gamma_2 e^{-\gamma_3 t}\|x(t)\| + \gamma_4 e^{-\gamma_5 t}$$

for all $x \in \mathbf{R}^n$ and $t \in \mathbf{R}_+$ where $\gamma_i \in \mathbf{R}_+$, $i = 1, \dots, 5$ and $\gamma_1 < \bar{\lambda}(P)\gamma_5$. Then the system state *x(t)* is uniformly bounded and exponentially converges to zero.

Proof. Denote $x^T(t)Px(t)$ by *V(t)* for brevity, then it follows that

$$\dot{V}(t) \le \frac{-\gamma_1}{\bar{\lambda}(P)}V(t) + \gamma_2 e^{-\gamma_3 t}\frac{V^{1/2}(t)}{\underline{\lambda}^{1/2}(P)} + \gamma_4 e^{-\gamma_5 t}$$

$$\doteq -\gamma_1' V(t) + \gamma_2' e^{-\gamma_3 t} V^{1/2}(t) + \gamma_4 e^{-\gamma_5 t}$$

or

$$V(t) \le e^{-\gamma_1' t}V(0) + \int_0^t e^{-\gamma_1'(t-\tau)}\left(\gamma_2' e^{-\gamma_3\tau}V^{1/2}(\tau) + \gamma_4 e^{-\gamma_5\tau}\right)d\tau$$

This expression can be arranged for the use of a Bihari-type inequality [15]

$$e^{-\gamma_1' t}V(t) \le V(0) + \frac{\gamma_4}{\gamma_1' - \gamma_5}\left(e^{(\gamma_1' - \gamma_5)t} - 1\right) + \gamma_2' \int_0^t e^{\left(\frac{1}{2}\gamma_1' - \gamma_3\right)\tau}\left[e^{\gamma_1'\tau}V(\tau)\right]^{1/2} d\tau$$

and for $\gamma_1' < \gamma_5$, it yields

$$e^{\dot{\gamma}_1 t} V(t) \leq \frac{1}{2}\left[2\left(V(0) + \frac{\gamma_4}{\gamma_5 - \dot{\gamma}_1} \right)^{1/2} + \int_0^t \dot{\gamma}_2 e^{\left(\frac{1}{2}\dot{\gamma}_1 - \gamma_3\right)\tau} d\tau \right]^2$$

and finally

$$\| x(t) \| \leq \frac{e^{-\frac{\dot{\gamma}_1}{2}t}}{\underline{\lambda}^{1/2}(P)} \left[2x^T(0) P x(0) + \frac{2\gamma_4}{\gamma_5 - \dot{\gamma}_1} \right]^{1/2} + \dot{\gamma}_2 \frac{\left(e^{-\gamma_3 t} - e^{\frac{-\dot{\gamma}_1}{2}t} \right)}{\underline{\lambda}^{1/2}\left(\frac{\dot{\gamma}_1}{2} - \gamma_3 \right)}$$

which shows the uniform bound and exponential stability.

3. Observer Design

Consider the nonlinear system given below

$$\dot{x} = F(x) + B[u + G(x, t)] \quad , \quad x(0) = x_0 \tag{1}$$

where $x \in \mathbf{R}^n$ is the state, $u \in \mathbf{R}^m$ the input, $F : \mathbf{R}^n \to \mathbf{R}^n$ is a nonlinearity with known form that satisfies $f(0) = 0$ and

$$\| F(x_1) - F(x_2) - A(x_1 - x_2) \| \leq b_A \| x_1 - x_2 \| \tag{2}$$

for some $A \in \mathbf{R}^{n \times n}$ for all $x_1, x_2 \in \mathbf{R}^n$. F therefore is a sector type nonlinearity, Ax representing the nominal linear model and b_A the maximum deviation from this model. $G : \mathbf{R}^n \times \mathbf{R}_+ \to \mathbf{R}^m$ represents the uncertain dynamics in which all the effects of unmodelled dynamics, parameter perturbations, disturbance signals are lumped. The nonlinearities are to be such that there exists a unique solution to (1) for a given x_0. The measurement equation is given by

$$y = Cx \tag{3}$$

with $y \in \mathbf{R}^p$.

In order to guarantee the convergence of our observer and stability of the controlled system, we will impose the following conditions on the system :

C1) There exists a matrix $P = P^T > 0$ such that

$$B^T P (I - C^{\#} C) = 0 \tag{4}$$

where P also satisfies the linear equation

$$(I - C^{\#} C)(A^T P + PA + Q_0)(I - C^{\#} C) = 0 \tag{5}$$

for some $Q_0 = Q_0^T > 0$.

C1′) There exists a matrix $P = P^T > 0$ that satisfies **C1** and it also is a solution to

$$(I - BB^{\#})(P^{-1} A^T + A P^{-1} + P^{-1} Q_c P^{-1})(I - BB^{\#}) = 0 \tag{6}$$

where $Q_c = Q_c^T > 0$.

C2) The uncertainty $G(x, t)$ satisfies

$$\| G(x, t) \|_s \leq b_G(y, t) < \infty \tag{7}$$

with $b_G : \mathbf{R}^p \times \mathbf{R}_+ \to \mathbf{R}_+$, for all $x \in \mathbf{R}^n$, $t \in \mathbf{R}_+$.

We start with the observer part of the dynamic compensator

$$\dot{\hat{x}} = F(\hat{x}) + Bu + S_0(y, \hat{x}) + K^0(y - C\hat{x}) \tag{8}$$

which is an identity observer, for the state given by (1). The term S_0 will be specified shortly. This results in an observation error $e \doteq x - \hat{x}$ dynamics :

$$\dot{e} = A_0 e + F(x) - F(\hat{x}) - Ae + BG - S_0 \tag{9}$$

where $A_0 = A - K^0 C$.

Introducing the Lyapunov function candidate $V(e) = e^T P e$, where $P>0$ satisfies condition **C1** , we obtain

$$\dot{V}(t) = e^T[A_0^T P + PA_0]e + 2e^T P[F(x) - F(\hat{x}) - Ae] + 2e^T P[BG - S_0] \tag{10}$$

We start with the first term on the right side of equation (10). One can see via lemma 2 that condition **C1** provides the necessary and sufficient condition for the assignability of a solution $P > 0$ for the equation

$$(A - K^0 C)^T + P(A - K^0 C) = -Q_0 < 0 \tag{11}$$

by the choice of a gain matrix K^0 whose value (which is not unique) can be calculated according to the formula :

$$K^0 = P^{-1}\left[\left(I - \frac{1}{2}C^{\#}C\right)(A^T P + PA + Q_0) + S_k^0\right]C^{\#} + Z_0(I - CC^{\#}) \tag{12}$$

where S_k^0 is a skew symmetric matrix that satisfies $S_k^0(I - C^{\#}C) = 0$ and Z_0 is arbitrary. Consequently, the first term on the right side of (14) will yield

$$e^T[A_0^T P + PA_0]e = -e^T Q_0 e < 0 \tag{13}$$

Now, we consider the second term on the right side of (10). Due to assumption on F given by (2), we have

$$2e^T P[F(x) - F(\hat{x}) - Ae] \le 2\bar{\lambda}(P) b_A \|e\|^2 \tag{14}$$

$$\doteq \alpha_1(P)\|e\|^2$$

The last term in (10) will finally be taken up. It can be seen that identity (4) is the necessary and sufficient condition for the existence of a matrix $H \in \mathbf{R}^{m \times p}$ such that [16]

$$B^T P = HC \tag{15}$$

Therefore, assumption **C1** allows us to rewrite and manipulate in the following way

$$2e^T PBG = 2e^T C^T H^T G = 2(y - C\hat{x})^T H^T G \le 2\|y - C\hat{x}\| \, \|H\|_s b_G(y, t) \tag{16}$$

This indicates the way to choose our nonlinear gain S_0 to guarantee negative definiteness in equation (10). Let

$$S_0(y, \hat{x}) = \frac{P^{-1}C^T(y - C\hat{x}) b_G^2 \|H\|_s}{\|y - C\hat{x}\| b_G + \varepsilon_0 e^{-\beta_0 t}} \tag{17}$$

for some $\varepsilon_0, \beta_0 \in \mathbf{R}_+$ as is done in [17]. Note that this is a continuous gain which results in the upper bound for the Lyapunov derivative

$$\dot{V} \le -e^T Q_0 e + \alpha_1(P)\|e\|^2 + 2\|H\|_s \varepsilon_0 e^{-\beta_0 t} \tag{18}$$

Assuming that $\underline{\lambda}(Q_0) > \alpha_1(P)$ picking β_0 such that $\underline{\lambda}(Q_0) - \alpha_1(P) < \beta_0 \bar{\lambda}(P)$ and using the results of lemma 3, $\|e\|$ can be shown to satisfy

$$\| e(t)\| \le \left[\frac{2e^T(0)Pe(0)}{\underline{\lambda}(P)} + \frac{4\|H\|_s \varepsilon_0}{\underline{\lambda}(P)(\beta_0 - 2\lambda_D)} \right]^{1/2} \mathrm{e}^{-\lambda_D t} \quad , \quad t \ge 0 \tag{19}$$

$\lambda_D = [\underline{\lambda}(Q_0) - \alpha_1(P)] / 2\overline{\lambda}(P) > 0$ so, the theorem below sums up this development.

Theorem 1. The observation error *e(t)* resulting form the use of the observer (8) with the gain K^0 given by (12) and S_0 by (17) used for the model (1) and (3) will obey (19) if F satisfies (2), assumptions **C1** and **C2** are met and $2\lambda_D < \beta_0$ where $\lambda_D > 0$ is defined above with $\alpha_1(P)$ given by (14).

Note that the additional condition $2\lambda_D < \beta_0$ imposed by the theorem guarantees the exponential decay envelope in the error dynamics (19) by restricting the deviation b_A of the nonlinearity F from the nominal linear dynamics Ax.

4. Observer-Based Control

We will now investigate the effect of using the observer/output based control strategy

$$u = K^c\hat{x} + S_c(y) \tag{20}$$

where

$$S_c(y) = -\frac{Hyb_G^2}{\|Hy\| b_G + \varepsilon_c \mathrm{e}^{-\beta_c t}} \quad , \quad \varepsilon_c , \quad \beta_c \in \mathbf{R}_+ \tag{21}$$

in the system (1)

$$\dot{x} = A_c x - BK^c e + F(x) - Ax + B[G + S_c] \tag{22}$$

where $A_c = A + BK^c$ and the definition of the observation error is used. The Lyapunov function $V(x) = x^T P x > 0$ where P satisfies the condition **C1′** will be used. Its derivative evaluated along the motion of system (22) yields

$$\dot{V}(x) = x^T[A_c^T P + PA_c]x - 2x^T PBK^c e + 2x^T P(F(x) - Ax) + 2x^T PB[G + S_c] \tag{23}$$

Because of assumption **C1′** and using lemma 1, the desired *P>0* is assignable as a solution to

$$A_c^T P + PA_c = -Q_c < 0 \tag{24}$$

and the gain that should be used will be

$$K^c = -B^{\#}(P^{-1}A^T + AP^{-1} + P^{-1}Q_c P^{-1})\left(I - \frac{1}{2}BB^{\#}\right)P - B^{\#}S_k^c P + (I - B^{\#}B)Z_c \tag{25}$$

where the skew-symmetric S_k^c satisfies $(I - BB^{\#})S_k^c = 0$ and Z_c is arbitrary.

The second term yields

$$-2x^T PBK^c e \le 2\|x\| \, \|PBK^c\|_s \|e\| \le 2\|PBK^c\|_s \left[\frac{2e^T(0)Pe(0)}{\underline{\lambda}(P)} + \frac{4\|H\|_s \varepsilon_0}{\underline{\lambda}(P)(\beta_0 - 2\lambda_D)} \right]^{1/2} \mathrm{e}^{-\lambda_D t} \|x\| \tag{26}$$

where (19) has been employed.

Form the third term, we obtain

$$2x^T P(F(x) - Ax) \le 2\overline{\lambda}(P) b_A \|x\|^2 \tag{27}$$

The last term in (23), via the use of the condition **C1**, **C2** and equations (15) and (21), gives

$$2x^T PB[G+S_c] = 2x^T C^T H^T [G+S_c] \le 2\|Hy\|\, b_G - \frac{2\|Hy\|^2 b_G^2}{\|Hy\|\, b_G + \varepsilon_c e^{-\beta_c t}} \tag{28}$$

Therefore, substituting from (24), (26), (27) and (28) into (23), we obtain

$$\dot{V}(x) \le -(\underline{\lambda}(Q_c) - 2\bar{\lambda}(P) b_A)\|x\|^2 + 2\|PBK^c\|_s \left[\frac{2e^T(0)Pe(0)}{\underline{\lambda}(P)} + \frac{4\|H\|_s \varepsilon_0}{\underline{\lambda}(P)(\beta_0 - 2\lambda_D)}\right]^{1/2} e^{-\lambda_D t}\|x\| + 2\varepsilon_c e^{-\beta_c t} \tag{29}$$

which is in a suitable form for the use of lemma 3. Making the proper identifications, we obtain

$$\|x(t)\| \le \frac{e^{-\frac{\dot{\gamma}_1}{2}t}}{\underline{\lambda}^{1/2}(P)}\left[2x^T(0)Px(0) + \frac{2\gamma_4}{\gamma_5 - \dot{\gamma}_1}\right]^{1/2} + \frac{\dot{\gamma}_2\left(e^{-\gamma_3 t} - e^{-\frac{\dot{\gamma}_1}{2}t}\right)}{\underline{\lambda}^{1/2}(P)\left(\frac{\dot{\gamma}_1}{2} - \gamma_3\right)}$$

where $\dot{\gamma}_1 = (\underline{\lambda}(Q_c) - 2\bar{\lambda}(P)b_A)/\bar{\lambda}(P)$, $\dot{\gamma}_2 = 2\|PBK^c\|_s\left[2e^T(0)Pe(0) + \frac{4\|H\|_s\varepsilon_0}{\beta_0 - 2\lambda_D}\right]^{1/2} / \underline{\lambda}(P)$, $\gamma_3 = \lambda_D$, $\gamma_4 = 2\varepsilon_c$, and $\gamma_5 = \beta_c$ which gives rise to the following result:

Theorem 2. The dynamic compensator given by equations (8), (12), (17), (20), (21), and (25), renders the state of system (1) uniformly bounded and exponentially convergent to zero provided that assumptions **C1′**, **C2**, equation (2), and the two conditions on design parameters $0 < 2\lambda_D < \beta_0$, $0 < [\underline{\lambda}(Q_c) - 2\bar{\lambda}(P)b_A] < \bar{\lambda}(P)\beta_c$ are true.

5. A Generalization

Consider again system (1) with the new measurement equation

$$y = Cx + h(x,t) \tag{30}$$

where $h : \mathbf{R}^n \times \mathbf{R}_+ \to \mathbf{R}^p$ models physical phenomena like sensor bias, measurement noise (bounded or "pink" type), etc. We will only be interested in the state reconstruction based on measurements (30). To be able to rephrase our estimation problem so that the results we have developed can be applied, equations (1) and (30) are rewritten in the following way:

$$\dot{X} = \bar{A}X + \bar{F}(X) + \bar{B}[u + \bar{G}(X,t)] \tag{31}$$

$$y = \bar{C}X \tag{32}$$

where

$$X = \begin{bmatrix} x \\ y \end{bmatrix}, \quad \bar{A} = \begin{bmatrix} A & 0 \\ CA & 0 \end{bmatrix}, \quad \bar{F}(X) = \begin{bmatrix} F - Ax \\ C(F - Ax) \end{bmatrix}$$

$$\bar{B} = \begin{bmatrix} B \\ CB \end{bmatrix}, \quad \bar{G}(X,t) = \begin{bmatrix} BG \\ CBG + \dot{h} \end{bmatrix}, \quad \bar{C} = [0 \quad I] \tag{33}$$

In order to guarantee the same convergence properties, the following modifications of the assumptions are proposed:

C1″)

$$(\overline{P}_{11}+\overline{P}_{12}C)B = 0 \tag{34}$$

where $\overline{P}_{11}\in \mathbf{R}^{n\times n}$, $\overline{P}_{11}=\overline{P}_{11}^{T}>0$ and $\overline{P}_{12}\in \mathbf{R}^{n\times p}$ are sub-matrices in $\overline{P}\in \mathbf{R}^{(n+p)\times(n+p)}$ such that

$$\overline{P} = \begin{bmatrix} \overline{P}_{11} & \overline{P}_{12} \\ \overline{P}_{12}^{T} & \overline{P}_{22} \end{bmatrix} > 0 \tag{35}$$

$$A^{T}(\overline{P}_{11}+\overline{P}_{12}C)^{T}+(\overline{P}_{11}+\overline{P}_{12}C)A+\overline{Q}_{0}^{11} = 0 \tag{36}$$

$\overline{Q}_{0}^{11}\in \mathbf{R}^{n\times n}$ being the first sub block of the matrix $\overline{Q}_{0}\in \mathbf{R}^{(n+p)\times(n+p)}$, $\overline{Q}_{0}=\overline{Q}_{0}^{T}>0$.

The following result can be proven based on theorem 1 :

Corollary 1. The estimation error remains uniformly bounded and it exponentially converges to zero if the observer

$$\dot{\hat{X}} = \overline{F}(\hat{X})+\overline{B}u+\overline{S}_{0}(y,\hat{X})+\overline{K}^{0}(y-\overline{C}\hat{X}) \tag{37}$$

is used for reconstructing the composite system state $X\in \mathbf{R}^{n+p}$ given by (31), where

$$\overline{S}_{0}(y,\hat{X}) = -\frac{\begin{bmatrix} \overline{P}_{12}^{-1} \\ \overline{P}_{22}^{-1} \end{bmatrix}(y-\overline{C}\hat{X})B_{\overline{G}}^{2}\|B^{T}(\overline{P}_{12}+C^{T}\overline{P}_{22})\|_{s}}{\|y-\overline{C}\hat{X}\|\,b_{\overline{G}}+\varepsilon_{0}e^{-\beta_{0}t}} \tag{38}$$

$$b_{\overline{G}} \geq \|\overline{G}(X,t)\|_{s}$$

and

$$\overline{K}^{0} = P^{-1}\left[\text{b. diag}\left(I,\frac{1}{2}I\right)\left(\overline{A}^{T}\,\overline{P}+\overline{P}\,\overline{A}+\overline{Q}_{0}\right)+\overline{S}_{k}^{0}\right]\begin{bmatrix}0\\I\end{bmatrix} \tag{39}$$

for a skew symmetric $\overline{S}_{k}^{0}\in \mathbf{R}^{(n+p)\times(n+p)}$ that satisfies

$$\overline{S}_{k}^{0} = \text{b. diag}\left(0,\overline{S}_{k}^{22}\right)\quad,\quad \overline{S}_{k}^{22}\in \mathbf{R}^{p\times p}\qquad \overline{S}_{k}^{22} = -\left(\overline{S}_{k}^{22}\right)^{T} \tag{40}$$

under the condition **C1″**, (2) and

$$0 < \underline{\lambda}(\overline{Q}_{0})-2\overline{\lambda}(\overline{P})\left(1+\|\overline{C}\|_{s}^{2}\right)^{1/2}b_{A} < \overline{\lambda}(P)\beta_{0} \tag{41}$$

Proof. The proof of this result follows from theorem 1 by proper substitutions. For example, the first term on the right side of (38)

$$\begin{bmatrix} \overline{P}_{12}^{-1} \\ \overline{P}_{22}^{-1} \end{bmatrix} = \overline{P}^{-1}C^{T}$$

and the last term in the numerator equals

$$\|B(\overline{P}_{12}+C^{T}\overline{P}_{22})\|_{s} = \|H\|_{s}$$

that follows from the new matching condition

$$\overline{B}^{T}\,\overline{P} = H\overline{C}$$

which is guaranteed by (34) which is equivalent to (4) with B, P, and C replaced by their respective new values $\overline{B}$, $\overline{P}$, and $\overline{C}$. (36) is equivalent to

$$(I-\overline{C}^{\#}\,\overline{C})\left(\overline{A}^{T}\,\overline{P}+\overline{P}\,\overline{A}+\overline{Q}_0\right)(I-\overline{C}^{\#}\,\overline{C}) = 0$$

so the existence of $\overline{K}^0$ written in simplified form in (39) that satisfies

$$(\overline{A}-\overline{K}^0\,\overline{C})^T+\overline{P}(\overline{A}-\overline{K}^0\,\overline{C}) = -\overline{Q}_0 < 0$$

can be guaranteed. Due to the specific form of $\overline{C}$ matrix, there is no more arbitrary term in the gain expression (39) and $\overline{S}_k^0$ takes its form in (40). The new terms in (41) arise due to the new bound on the expression

$$2\overline{e}^T\,\overline{P}[\overline{F}(X)-\overline{F}(\hat{X})-\overline{A}\,\overline{e}] \le 2\overline{\lambda}(\overline{P})(1+\|C\|_s^2)^{1/2}b_A\|\overline{e}\|^2$$

where $\overline{e}$ is the observation error for the composite system.

6. Simulation Results

Example : Consider the system in [8] modeled by the relationships:

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}0\\ -\sin x_1\end{bmatrix}$$

$$y = x_1+x_2$$

Following theorem 1, and taking

$$F(x)=Ax \quad , \quad A = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix} \quad , \quad G(x) = \begin{bmatrix}0\\ -\sin x_1\end{bmatrix} \quad \text{and} \quad C = [1 \quad 1]$$

will satisfy assumption C1 , C1′ and C2 for the following P , Q_0 , and Q_c :

$$P = \begin{bmatrix}2 & 1\\ 1 & 1\end{bmatrix} \qquad Q_0 = Q_c = \begin{bmatrix}6 & 3\\ 3 & 2\end{bmatrix}$$

b_A can be taken as zero and $\|G\|_s \le 1$. For the observer, H is found to be 1 . K^0 with $S_k^0=0$ gives $K^0=[1 \quad 1]^T$. This will result in $\lambda_D>0$ and a choice of $\beta_0>0.1506$ satisfies $2\lambda_D<\beta_0$ condition. K^c is chosen as $[-1.5 \quad -1.5]^T$. The choice of $\beta_c>0.2590$ will also satisfy the design parameter constraint. Therefore, the closed loop system given by this example will be uniformly bounded and exponentially convergent to zero.

For simulation purposes, we have applied a first order Euler approximation with sampling period of 0.02 to the continuous system. The initial conditions are : $x_1(0)=\frac{\pi}{2}$ and $x_2(0)=0$. $\varepsilon_0=\varepsilon_c$ has been chosen as 0.01. *Figure 1* shows the propagation of x_1 with and without this controller. *Figure 2* represents the propagation of x_2 with and without this controller. As it can be inferred from these figures both states exponentially converge to zero.

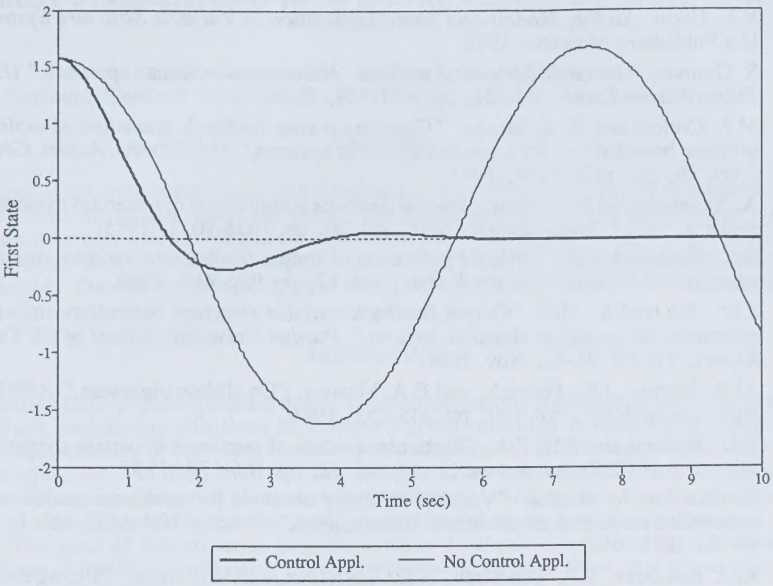


Figure 1

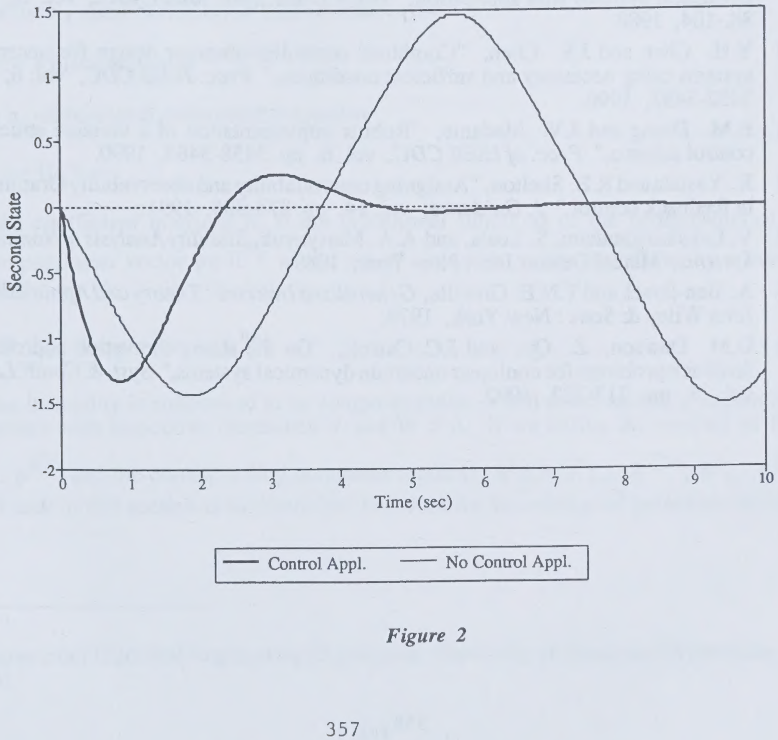


Figure 2

REFERENCES

[1] V.I. Utkin, *Sliding Models and Their Application in Variable Structure Systems*, Mir Publisher : Moscow, 1978.

[2] S. Gutman, "Uncertain dynamical systems - A Lyapunov min-max approach," *IEEE Trans. Autom. Contr.*, vol 24, pp. 437-443, 1979.

[3] M.J. Corless and G. Leitmann, "Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems," *IEEE Trans. Autom. Contr.*, vol. 26, pp. 1139-1144, 1981.

[4] A. Steinberg and M. Corless, "Output feedback stabilization of uncertain dynamical systems," *IEEE Trans. Autom Contr.*, vol. 30, pp. 1025-1027, 1985.

[5] B.S. Heck and A.A. Ferri, "Application of output feedback to variable structure systems," *J. Guidance Contr. & Dyn.*, vol. 12, pp. 932-935, 1989.

[6] S.H. Zak and S. Hui, "Output feedback variable structure controllers and state estimators for uncertain dynamic system," *Purdue University School of EE Tech. Report*, TR-EE 91-46, Nov. 1991.

[7] J.J.E. Slotine, J.K. Hedrick, and E.A. Misawa, "On sliding observers," *ASME J. Dyn. Syst. & Meas.*, vol. 109, pp. 245-252, 1986.

[8] B.L. Walcott and S.H. Zak, "State observation of nonlinear uncertain dynamical systems," *IEEE Trans. Autom. Contr.*, vol. 32, pp. 166-170, 1987.

[9] E. Yaz and A. Azemi, "Variable structure observer for nonlinear models with unbounded noise and measurement uncertainties," *Proc. of 10th ACC,* vol. 1, pp. 43-48, 1991.

[10] A.G. Bondarev, S.A. Bondarev, N.E. Kostyleva, and V.I. Utkin, "Sliding modes in systems with asymptotic state observers," *Automatika i Telemekhanika,* pp. 5-11, 1985.

[11] B.L. Walcott and S.H. Zak, "Combined observer-controller synthesis for uncertain dynamical systems with application," *IEEE Trans. Syst. Man Cyber.*, vol. 18, pp. 88.-104, 1988.

[12] Y.H. Chen and J.S. Chen, "Combined controller-observer design for uncertain systems using necessary and sufficient conditions," *Proc. IEEE CDC,* vol. 6, pp. 3452-3457, 1990.

[13] B.M. Diong and J.V. Medanic, "Robust implementation of a variable structure control scheme," *Proc. of IEEE CDC*, vol. 6, pp. 3458-3463, 1990.

[14] K. Yasuda and R.E. Skelton, "Assigning controllability and observability Gramianas in feedback control," *J. Guidance*, vol. 14, pp. 878-885, 1991.

[15] V. Lakshmikantham, S. Leela, and A.A. Martynyuk, *Stability Analysis of Nonlinear Systems*, Marcel Dekker Inc. : New York, 1989.

[16] A. Ben-Israel and T.N.E. Greville, *Generalized Inverses : Theory and Applications*, John Wiley & Sons : New York, 1974.

[17] D.M. Dawson, Z. Qu, and J.C. Carroll, "On the state observation and output feedback problems for nonlinear uncertain dynamical systems," *Syst. & Contr. Letts*, vol. 18, pp. 217-222, 1992.

ROBUST STABILIZATION OF INTERVAL PLANTS BY STATE SPACE METHODS

K. Grigoriadis, E. Yaz*, R. Skelton
Space Systems Control Laboratory, Purdue University, W. Lafayette, IN

Abstract

This paper is on output feedback stabilization and covariance control of continuous-time interval plants using state space methods. The technique is based on some linear algebra results on common solution of matrix equations. A finite step algorithm is given for this purpose, which is also shown to be applicable to robust stabilization problems for structured uncertainties. The results are also applied to some extensions of Kharitonov's results to the matrix case.

Introduction

An important topic in robust control theory is the simultaneous stabilization of multiple plants [1-4]. Such engineering situations arise when a given controller is required to stabilize the system under a variety of operating conditions characterized by different sets of coefficients for the state equations. An even more difficult problem is the stabilization of interval plants. In these situations, the number of plant models requiring stabilization by a fixed compensator is infinite in number, as the state space coefficients take values on a continuum between given limits. The goal of this paper is to present some algebraic conditions for simultaneous stabilization of multiple plants using static output feedback and to show how to use this result to solve the interval plant stabilization problem.

Covariance-Based Results

Consider the n_x-dimensional linear time-invariant system

$$\dot{x} = A(p)x + B(p)u + v \tag{1}$$

with the n_y-dimensional measurement equation

$$y = Cx + w \tag{2}$$

where the coefficient matrices A, B are multilinear functions of the components of the unknown parameter vector $p \in R^{n_p}$ whose set of allowable values is defined by a convex polyhedron,

$$\mathcal{P} \underline{\underline{\Delta}} \{p:\ \underline{\beta} \le p \le \bar{\beta},\ p \in R^{n_p}\} \tag{3}$$

where the inequality is understood to be componentwise. v and w are mutually uncorrelated white noises with respective intensities V and W > 0. If we define the vertices of P by $p^1, p^2, \ldots, p^{2^{n_p}}$ and the corresponding parameter matrices $A^1 \underline{\underline{\Delta}} A(p^1), \ldots, A^{2^{n_p}}$, $B^1, \ldots, B^{2^{n_p}}$, then our task in this section is to determine, based on the knowledge of parameter matrices evaluated at the vertices of $\mathcal{P}$, the assignability conditions of a common covariance matrix X > 0 by a static output feedback gain

---

* On leave from Electrical Engineering Department, University of Arkansas, Fayetteville, AR 72701.

$$u = Ky \tag{4}$$

and in case the assignability conditions are satisfied, to compute the necessary gain matrix K. Note that if covariance assignment is satisfied by a single gain K, then we achieve robust stabilization of the above interval plant according to the following result in [8]:

Lemma 1. Let the family of linear, autonomous system be defined by (1) with $u \equiv 0$, $v \equiv 0$, and (3). Then, A(p) is asymptotically stable for all $p \in \mathcal{P}$ if there exists a real symmetric matrix $\tilde{X} > 0$ such that

$$\tilde{X}A^i + A^{i^T}\tilde{X} < 0, \quad i = 1, ..., 2^{n_p}. \tag{5}$$

The use of the control in (4) results in the closed loop system

$$\dot{x} = \left[A^i + B^iKC\right]x + B^iKw + v \underset{=}{\Delta} A^i_K + B^iKw + v, \quad i = 1,...,2^{n_p} \tag{6}$$

which corresponds to the vertex parameter values. The common steady state second moment (we will call this covariance to conform with the established usage in [6-7]) of the state vector will satisfy

$$0 = A^i_K X + XA^{i^T}_K + B^iKWK^TB^{i^T} + V, \quad i = 1, ..., 2^{n_p} \tag{7}$$

We will first investigate the assignability of a positive definite covariance that solves (7) and we will also derive the feedback gains K to make this assignment.

Rearranging equation (7) yields

$$\begin{aligned} -(A^iX + XA^{i^T} + V) &= B^iKCX + XC^TK^TB^{i^T} + B^iKWK^TB^{i^T} \\ &= (B^iK + G)W(B^iK + G)^T - GWG^T \end{aligned} \tag{8}$$

where the square is completed on the right side with

$$G \underset{=}{\Delta} XC^TW^{-1} \tag{9}$$

based on the fact that W > 0. Upon rearrangement again, we obtain

$$-(A^iX + XA^{i^T} + V - XC^TW^{-1}CX) = (B^iK + G)W(B^iK + G)^T \tag{10}$$

where (9) is substituted. An assignable covariance must render the left side of (10) nonnegative definite with a rank not exceeding n_y since the right side has these properties. If this is true, we rewrite (10) as

$$L^iL^{i^T} = (B^iK + G)W^{1/2}W^{T/2}(B^iK + G)^T \tag{11}$$

where $L^i \in R^{n_x x n_y}$, $W^{1/2} \in R^{n_y x n_y}$ is nonsingular. But (11) is equivalent to

$$L^iU^i = (B^iK + G)W^{1/2} \tag{12}$$

for some orthogonal matrix $U^i \in R^{n_y x n_y}$ [11], or

$$B^iK = L^iU^iW^{-1/2} - G\underline{\underline{\Delta}}D^i, \; i = 1,\ldots,2^{n_p} \tag{13}$$

so the gain matrix K should simultaneously be able to satisfy all 2^{n_p} equations. The solutions of such common matrix equations are considered in [5] and from the results of that work we obtain that the existence conditions and the value of the common solution of equation (13) can be expressed as follows.

__Lemma 2.__ Define

$$E^1 = B^1, \; F^1 = D^1, \; H^1 = (B^1)^{\dagger}D^1, \; J^1 = I - (B^1)^{\dagger}B^1 \tag{14}$$

$$E^i = B^iJ^{i-1}, \; F^i = D^i - B^iH^{i-1}, \; H^i = H^{i-1} + J^{i-1}(E^i)^{\dagger}F^i$$

$$J^i = J^{i-1}[I - (E^i)^{\dagger}E^i], \; i = 2, 3, \ldots, 2^{n_p} \tag{15}$$

Then 2^{n_p} equations in (13) have a common solution K if and only if

$$[I - E^i(E^i)^{\dagger}]F^i = 0, \; i = 1, \ldots, 2^{n_p} \tag{16}$$

in which case the common solution is

$$K = H^{2^{n_p}} + J^{2^{n_p}}Z \tag{17}$$

where Z is arbitrary.

The above result makes it possible to both test the assignability of a given covariance matrix and also to find the necessary orthogonal matrices U^i in the control formula in a recursive way as shown below. Let i = 1, then from (14)-(16), the first assignability condition is

$$\left[I - B^1(B^1)^{\dagger}\right]\left(L^1U^1W^{-1/2} - G\right) = 0 \tag{18}$$

which when rearranged, squared and upon substitution from (11) yields

$$\left[I - B^1(B^1)^{\dagger}\right]\left\{L^1U^1(U^1)^T(L^1)^T - GW(G)^T\right\}\left[I - B^1(B^1)^{\dagger}\right] = 0 \tag{19}$$

$$\Leftrightarrow \quad \left[I - B^1(B^1)^{\dagger}\right]\left[A^1X + X(A^1)^T + V\right]\left[I - B^1(B^1)^{\dagger}\right] = 0 \tag{20}$$

which turns out to be the assignability condition of a given state covariance X to the system (A^1, B^1) (see [6]). One can also deduce from equation (18) that the orthogonal matrix U^1 can be solved as follows

$$U^1 = \mathcal{V}_1 \text{block diag}(I_p, \; \overline{U}^1)\mathcal{V}_2^T \tag{21a}$$

where $\mathcal{V}_1$ and $\mathcal{V}_2$ are found from the singular value decompositions

$$[I - B^1(B^1)^\dagger]L^1 = \mathcal{U}_1\Sigma_1\mathcal{V}_1^T \tag{21b}$$

$$[I - B^1(B^1)^\dagger]GW^{1/2} = \mathcal{U}_1\bar{\Sigma}_1\mathcal{V}_2^T, \quad \bar{\Sigma}_1 = [\Sigma_1,\ 0] \tag{21c}$$

with $\Sigma_1 \in R^{p\times p}$ being a diagonal matrix having all nonzero singular values of the matrices in question and $\overline{U}^1$ is arbitrary orthogonal.

At this point, we use an inductive argument. Suppose, up through the $j-1^{st}$ stage the assignability conditions in (16) are satisfied. At the j^{th} stage $(1 < j \leq 2^{n_p})$, we have

$$\left[I - E^j(E^j)^\dagger\right]F^j = 0 \tag{22}$$

or

$$\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]\left(D^j - B^jH^{j-1}\right) = 0 \tag{23}$$

It is easy to show that J^{j-1} is a function of only $B^1,\ldots,B^{j-1}$ from (14) and (15), whereas $D^j = L^jU^jW^{-1/2} - G$ and H^{j-1} is dependent on known or predetermined quantities $(B^1,\ldots,B^{j-1},\ L^1,\ldots,L^{j-1},\ U^1,\ldots,U^{j-1})$ by induction hypothesis. Therefore, (23) yields

$$\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]L^jU^j = \left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]\left(G + B^jH^{j-1}\right)W^{1/2} \tag{24}$$

or

$$\begin{aligned}&\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]\left[L^jU^jU^{j^T}L^{j^T} - \left(G + B^jH^{j-1}\right)W\left(G + B^jH^{j-1}\right)^T\right]\\&\qquad\times\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right] = 0\end{aligned}$$

Substituting from (9)-(11), we obtain

$$\begin{aligned}&\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]\left[A^jX + XA^{j^T} + V + B^jH^{j-1}CX + XC^TH^{j-1^T}B^{j^T}\right.\\&\qquad\qquad\left. + B^jH^{j-1}W(H^{j-1})^TB^{j^T}\right]\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right] = 0\end{aligned} \tag{25}$$

which can be used to check the j^{th} assignability condition independent of the orthogonal matrix U^j. To find U^j we can just use equation (24) to obtain

$$U^j = \mathcal{V}_1^j \text{ block diag}(I_{p_j},\ \overline{U}^j)\mathcal{V}_2^{jT} \tag{26a}$$

where $\mathcal{V}_1^j$ and $\mathcal{V}_2^j$ are found from

$$\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]L^j = \mathcal{U}_1^j\Sigma_1^j\mathcal{V}_1^j \tag{26b}$$

$$\left[I - B^jJ^{j-1}(B^jJ^{j-1})^\dagger\right]\left(G^j + B^jH^{j-1}\right)W^{1/2} = \mathcal{U}_1^j\bar{\Sigma}_1^j\mathcal{V}_2^{jT},\ \bar{\Sigma}_1^j = \left[\Sigma_1^j,\ 0\right] \tag{26c}$$

and $\Sigma_1^j \in R^{p_j x p_j}$ is a diagonal matrix containing all nonzero singular values of the left hand side matrices in (26b) and (26c), and $\overline{U}^j$ is arbitrary orthogonal.

Theorem 1. A common covariance matrix X > 0 is assignable to the 2^{n_p} equations in (7) by a single gain matrix K if and only if there exists X > 0 such that the left side of equation (10) is nonnegative definite with rank not exceeding n_y and X satisfies equations (20) and (25) for i = $2,\ldots,2^{n_p}$. If a certain X > 0 is assignable, then the assignment can be realized by the gain matrix (17) parametrized by an arbitrary Z and the orthogonal matrices U^i that can be found from (21a)-(21c) for i = 1 and (26a)-(26c) for i = $2,\ldots,2^{n_p}$ in the recursive fashion described above.

At this point, we assume that V > 0 to avoid dealing with the controllability property of the closed loop system (6) in the coming discussion, although this was not necessary for covariance assignment. This guarantees that there exists a symmetric matrix X > 0 that satisfies

$$A_K^i X + X A_K^{i^T} < 0 \tag{27}$$

What we need, however, is the satisfaction of the dual equation (5):

$$(A_K^i)^T \tilde{X} + \tilde{X}(A_K^i) < 0 \tag{28}$$

if we want to guarantee the robustness property promised by Lemma 1. So, we use the well-known result:

Lemma 3. There exists a solution $X = X^T > 0$ to (27) if and only if there exists a solution $\tilde{X} = \tilde{X}^T > 0$ to (28).

So, based on this equivalence of equations (27) and (28) given by Lemma 3 and the results of Theorem 1, we can now state our stability robustness result.

Theorem 2. Let V, W > 0 and the assignability conditions of Theorem 2 hold, then the controller described in Theorem 2 will render the family of interval plants given by (1), (2) robustly asymptotically stable for all parameter values $p \in \mathcal{P}$.

An important application of this result is to deterministic systems with no additive noise terms $v \equiv 0$, $w \equiv 0$ (although we have included the noise in the system description to motivate the covariance interpretation of the solution to equation (7)). In this case one needs to choose matrices $V \in R^{n_x x n_x}$, $V = V^T > 0$, $W \in R^{n_y x n_y}$, $W = W^T > 0$, which are included in the parameterization of the robustly stabilizing controllers. Note that similar results can be obtained when C = C(p) is a multilinear function of the parameter vector p and B is constant.

Results for Structured Uncertainty

If more structure is introduced into the problem formulation, then stronger results can be obtained. In this section, we base our development on a result given in [9].

Lemma 4. Consider the family of linear systems given by (1) with $u \equiv 0$ and $v \equiv 0$. Then A(p) is asymptotically stable for any A belonging to

$$\mathcal{A} = \left\{ A(p):\ A(p) = \sum_{i=1}^{n_p} p_i A^i,\ \ p_i \ge 0,\ \ \sum_{i=1}^{n_p} p_i = 1 \right\} \tag{29}$$

if there exists $\tilde{X} = \tilde{X}^T > 0$ such that

$$A^{i^T}\tilde{X} + \tilde{X}A^i < 0 \tag{30}$$

One can see that the above lemma allows us to deal with fewer numbers of parameters. In this case, a straightforward application of the above results will result in a failure since the solution will depend on the knowledge of unknown parameters p_i. So, we change our angle of attack and tackle the problem of assigning a solution $X = X > 0$ to the following equation

$$0 = A_K^i X + X A_K^{i^T} + B^i K W K^T B^{i^T} + V^i \tag{31}$$

without trying to attach a covariance interpretation to it. Therefore $V^i \in R^{n_x x n_x}$ and $W \in R^{n_y x n_y}$ are arbitrary symmetric matrices with $V^i \ge 0$ and $W > 0$. This time, we would like to simultaneously stabilize the family of interval plants

$$\dot{x} = \left(\sum_{i=1}^{n_p} p_i A^i \right) x + \left(\sum_{i=1}^{n_p} p_i B^i \right) u \tag{32}$$

$$y = \left(\sum_{i=1}^{n_p} p_i C^i \right) x \tag{33}$$

where the p_i's are specified in (29), by a single static output feedback controller (4). First, we will assume that there is no uncertainty in the input matrix, so consider

$$\dot{x} = \left(\sum_{i=1}^{n_p} p_i A^i \right) x + Bu \tag{34}$$

together with (33). Later on, output matrix will be assumed known and description (32) will be used for the plant. For the first problem, we use the design equation

$$0 = (A^i + BKC^i)X + X(A^i + BKC^i)^T + BKWK^TB^T + V^i,\ i = 1,\ldots,n_p \tag{35}$$

for which the assignability of a given $X = X^T > 0$ and its assignment by the gain matrix K will be considered. Then, it will be easy to show how (35) implies the results of Lemma 4. But the first part of this task is easy since equation (35) is the same as equation (7) with $B^i = B$, $i = 1,\ldots,n_p$. Therefore, it follows form Theorem 1 that similar results are valid.

Corollary 1. A common solution $X = X^T > 0$ to n_p equations in (35) is assignable under the same necessary and sufficient conditions given in Theorem 1 with 2^{n_p} replaced by n_p and

$B^i = B$, $i = 1,\ldots,n_p$. In this case, the same set of gain formulae (17), where U^i, $i = 1,\ldots,n_p$ calculated in exactly the same way, will assign the required solution.

If we let $V^i > 0$ now, one can see that Lemma 4 can be used together with Lemma 3 to state:

Theorem 3. Let $V^i > 0$ and the assignability conditions of Corollary 1 be satisfied. Then the controller (17) described in detail in Theorem 1 will stabilize all the closed loop systems

$$\dot{x} = \sum_{i=1}^{n_p} p_i (A^i + BKC^i)x, \quad i = 1,\ldots,n_p \tag{36}$$

and therefore stabilize the family of interval plants in the convex polytope (29) with $A(p)$ replaced by $\sum_{i=1}^{n_p} p_i (A^i + BKC^i)$.

We can now do the same thing for system (32) with the measurement equation

$$y = Cx \tag{37}$$

Corollary 2. A common solution $X = X^T > 0$ to the following set of equations

$$0 = (A^i + B^i KC)X + X(A^i + B^i KC)^T + B^i KWK^T B^{i^T} + V^i, \quad i = 1,..,n_p \tag{38}$$

is assignable if and only if the same conditions of Theorem 1 hold true with 2^{n_p} replaced by n_p and $C^i = C$, $i = 1,\ldots,n_p$, in which case the same gain formulae will assign the desired solution.

Theorem 4. Let $V^i > 0$ and the assignability conditions of X in (38) are satisfied, then the same controller as above will stabilize all the closed loop plants with the system matrix $\sum_{i=1}^{n_p} p_i (A^i + B^i KC)$, $p_i \geq 0$, $\sum_{i=1}^{n_p} p_i = 1$.

Although it has been easier to solve this problem without using a valid covariance equation, one can also view the above solutions as assigning a common covariance to

$$\dot{x} = (A^i + BKC^i)x + BKw + v, \quad i = 1,\ldots,n_p \tag{39}$$

when B is constant and

$$\dot{x} = (A^i + B^i KC)x + B^i Kw + v, \quad i = 1,\ldots,n_p \tag{40}$$

when C is constant with the intensities of v and w being V and W, respectively. Note that this identification can lead to the use of covariance equations (35) and (38) for systems (39) and (40), respectively, which is permitted for the special case of the uncertainty structure (29).

Kharitonov-Based Results

We now assume a system with parameter matrices being in the special forms

$$\dot{x} = A_c x + B_c u + v \tag{41}$$

$$y = Cx + w \tag{42}$$

$$A_c = \begin{bmatrix} 0 & I_{n_x-1} \\ -\beta_0 \ldots & -\beta_{n_x-1} \end{bmatrix}, \quad B_c = \begin{bmatrix} 0 \\ b \end{bmatrix}, \quad b \in R^{1xm} \tag{43}$$

where $0 \in R^{n_x-1}$ and the rest of the description being the same. The limitations of this structure in modeling are pointed out in [10] and this special structure is used to facilitate the application of Kharitonov's robust stability results for polynomials to state-space models. In particular, the following result is shown:

Lemma 5. Let $K \in R^{n_u x n_y}$ be given and consider

$$\mathcal{A}_K = \{A_c + B_c KC: \ A_c \in \mathcal{A}\} \tag{44}$$

where

$$\mathcal{A} = \{A_c: \ \underline{\beta}_i \leq \beta_i \leq \overline{\beta}_i, \ i = 0,\ldots,n_x\} \tag{45}$$

Then, every matrix in $\mathcal{A}_k$ is stable if and only if the four matrices $A_K^i = A_c^i + B_c KC$, $i = 1,\ldots,4$ are stable where

$$A_c^1 = \begin{bmatrix} 0 & & I_{n_x-1} & \\ \cdots - \underline{\beta}_{n_x-4} & - \underline{\beta}_{n_x-3} & -\overline{\beta}_{n_x-2} & - \overline{\beta}_{n_x-1} \end{bmatrix}$$

$$A_c^2 = \begin{bmatrix} 0 & & I_{n_x-1} & \\ \cdots - \overline{\beta}_{n_x-4} & - \underline{\beta}_{n_x-3} & \underline{-\beta}_{n_x-2} & - \overline{\beta}_{n_x-1} \end{bmatrix}$$

$$A_c^3 = \begin{bmatrix} 0 & & I_{n_x-1} & \\ \cdots - \overline{\beta}_{n_x-4} & - \overline{\beta}_{n_x-3} & \underline{-\beta}_{n_x-2} & - \underline{\beta}_{n_x-1} \end{bmatrix}$$

$$A_c^4 = \begin{bmatrix} 0 & & I_{n_x-1} & \\ \cdots - \underline{\beta}_{n_x-4} & - \overline{\beta}_{n_x-3} & -\overline{\beta}_{n_x-2} & - \underline{\beta}_{n_x-1} \end{bmatrix} \tag{46}$$

To be able to rephrase the robust stabilization problem in terms of covariance assignment, we will define four covariance equations:

$$0 = (A_c^i + B_c KC)X^i + X^i(A_c^i + B_c KC) + B_c KWK^T B_c^T + V \tag{47}$$

where this time we do not need to assign the same covariance to these equations, since it is sufficient to make four different closed loop matrices asymptotically stable. So, suppose that in addition to stability we have some performance criteria -- which can be expressed as a set of covariance matrices $X^1,\ldots,X^4$ [6-7]. Again, we first tackle the assignability problem.

Corollary 3. A desired set of closed loop covariance matrices $X^i = (X^i)^T > 0$, $i = 1,\ldots,4$, is assignable to equations in (47) under the same necessary and sufficient conditions of Theorem 1 with the following replacements:

$$X \to X^i,\ A^i \to A_c^i,\ B^i \to B_c,\ C^i \to C \tag{48}$$

where $i = 1,\ldots,4$, and the same controller gain K with the obvious modifications indicated above will guarantee the assignment of these solutions.

Theorem 5. If the assignability conditions of Corollary 3 are satisfied, then the controller given by Corollary 3 will robustly stabilize the family of interval plants given by (41)-(45).

Specialization to Simultaneous Stabilization

The results of Theorem 1 can also be used for simultaneously stabilizing a (discrete) family of say n_p systems with the same controller. The importance of such a solution (e.g., in problems involving actuator and sensor failures) is emphasized in [1-4]. In this case also, we do not have to assign the same covariance to all members of the family but different covariances X^i, $i = 1,\ldots,n_p$ chosen based on different performance criteria can be utilized.

Theorem 6. If the assignment conditions of Theorem 1 are satisfied for all X^i, $i = 1,\ldots n_p$, then the controller given by that theorem for X replaced by X^i, will simultaneously stabilize a discrete family of n_p plants given by

$$\dot{x} = A^i x + B^i u + v^i \tag{49}$$

$$y = C^i x + w^i \tag{50}$$

Conclusions

Linear algebra results on the common solution of linear matrix equations have been used to derive algebraic conditions for robust stabilization of multiple and interval plants. A different interpretation of the results provides the required assignability conditions for a covariance matrix to be assignable to interval plants by a common output feedback controller. A parametrization of the corresponding feedback gains is also provided. The interval plant stabilization or covariance assignment is accomplished by a finite step procedure which is based on covariance assignment for a finite number of plants.

References

[1] M. Vidyasagar, Control System Synthesis: A Factorization Approach, MIT Press: Cambridge, 1985.

[2] B.K. Ghosh, "Approach to simultaneous system design: part II. Nonswitching gain and dynamic feedback compensation by algebraic geometric methods," SIAM J. Contr. & Optim., v. 26, pp. 919-963, 1988.

[3] M. Vidyasagar, "State space interpretation of simultaneous stabilization," IEEE Trans. Autom. Contr., v. 33, pp. 506-508, 1988.

[4] P.R. Khargonekar, A.M. Pascoal, and R. Ravi, "Strong, simultaneous, and reliable stabilization of finite-dimensional linear time-varying plants," IEEE Trans. Autom. Contr., v. 33, pp. 1158-1161, 1988.

[5] G.L. Morris and P.L. Odell, "Common solutions for n matrix equations with applications," J. Assoc. Computing Machinery, v. 15, pp. 272-274, 1968.

[6] A. Hotz and R.E. Skelton, "Covariance control theory," Int. J. Contr., v. 46, pp. 13-32, 1987.

[7] R.E. Skelton and M. Ikeda, "Covariance controllers for linear continuous time systems," Int. J. Contr., v. 49, pp. 1773-1785, 1989.

[8] H.P. Horisberger and P.R. Belanger, "Regulators for linear, time invariance plants with uncertain parameters," IEEE Trans. Autom. Contr., v. 21, pp. 705-711, 1976.

[9] H. Kokame, H. Kida, and T. Mori, "A polytopic quadratic Lyapunov functions approach to stability of a matrix polytope," Proc. 29th CDC, pp. 3500-3505, Honolulu, HI, 1990.

[10] D.S. Bernstein and W.M. Haddad, "Robust controller synthesis using Kharitanov's theorem," IEEE Trans. Autom. Contr., v. 37, pp. 129-132, 1992.

[11] C.R. Rao and S.K. Mitra, Generalized Inverse of Matrices and Its Applications, Wiley and Sons: New York, 1971.

IMPROVED SUFFICIENT CONDITIONS FOR THE STABILITY OF INTERVAL MATRICES*

Kaining Wang Anthony N. Michel
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556, U.S.A.

I. Introduction

The stability of interval matrices is of great current interest (see e.g., [1]). It has been shown that Lyapunov's second method constitutes a powerful tool in stability studies of such matrices.

In [2], a sufficient condition is established for the Hurwitz stability of real $n \times n$ interval matrices which involves examining $2^{n(n-1)/2}$ corners of a certain $n \times n$ interval matrix. In the present paper, we establish a sufficient condition for the Hurwitz stability of real $n \times n$ interval matrices which requires a check of only 2^{n-1} corners of the aforementioned interval matrix. Moreover, these corners form a subset of the above $2^{n(n-1)/2}$ corners.

In the next section we provide necessary notation and in Section 3 we present the main results. Due to space limitations, the proofs and the generalization of the results in Section 3 have been omitted.

II. Notation

Let R^n denote real n-space. If $x \in R^n$, then $x^T = (x_1, \cdots, x_n)$ denotes the transpose of x and if $x, y \in R^n$, then $x \leq y$ signifies $x_i \leq y_i$, $x < y$ signifies $x_i < y_i$, and $x > 0$ denotes $x_i > 0$ for all $i = 1, \cdots, n$.

Let $A = [a_{ij}]_{nxn}$ denote a real $n \times n$ matrix and let A^T denote the transpose of A.

We let I and J denote subsets of $\{1, 2, \cdots, n\}$ satisfying the following properties:

(i) $I \cup J = \{1, 2, \cdots, n\}$

(ii) $I \cap J = \phi$.

For $A = [a_{ij}]_{nxn}$ and I, J as specified above, we let A_{IJ} denote the submatrix of A given by

$$A_{IJ} \triangleq \begin{bmatrix} a_{i_1 j_1} & \cdots & a_{i_1 j_{n-s}} \\ \vdots & & \vdots \\ a_{i_s j_1} & \cdots & a_{i_s j_{n-s}} \end{bmatrix}. \tag{2.1}$$

For $B = [b_{ij}]_{nxn}$ and $C = [c_{ij}]_{nxn}$ and $C = [c_{ij}]_{nxn}$ satisfying $b_{ij} \leq c_{ij}$ for all $1 \leq i, j \leq n$, we define the $n \times n$ *interval matrix* $[B, C]$ by

$$[B, C] = \{A = [a_{ij}]_{nxn} : b_{ij} \leq a_{ij} \leq c_{ij},\ 1 \leq i, j \leq n\}.$$

A matrix $A = [a_{ij}]_{nxn} \in [B, C]$ for which $a_{ij} = b_{ij}$ or $a_{ij} = c_{ij}$, for all $i, j = 1, \ldots, n$, is called a *corner* of the interval matrix $[B, C]$.

If for all $A \in [B, C]$, A is Hurwitz stable (i.e., all eigenvalues of A have negative real parts), then we say that $[B, C]$ is *Hurwitz stable.*

---

*This work was supported by the National Sciences Foundation under Grant ECS91-07728.

III. Main Results

The following theorem improves the results established in [2] considerably.

Theorem 1. The interval $n \times n$ matrix $[B, C]$ is Hurwitz stable if *every* matrix S_{IJ}, given by

$$S_{IJ} = \begin{bmatrix} C_{II} & B_{IJ} \\ B_{JI} & C_{JJ} \end{bmatrix} + \begin{bmatrix} C_{II} & B_{IJ} \\ B_{JI} & C_{JJ} \end{bmatrix}^T \tag{3.1}$$

is negative definite, where I, J are specified in Section 2, and C_{II}, C_{JJ}, B_{IJ} and B_{JI} are submatrices of C and B, respecively, following the convention (2.1).

Remark 1. Previous results (see [2] or [1, Theorem 20]) require verification of the definiteness of $2^{(n-1)n/2}$ corners of the interval matrix $[B + B^T, C + C^T]$.

The decomposition $x = \begin{bmatrix} x_I \\ x_J \end{bmatrix}, x_I \geq 0,\ x_J < 0$ (refer to the proof of Theorem 1), induces a partition of R^n which consists of 2^n components. Therefore, there are 2^n different matrices S_{IJ} (defined by (3.1)). Since S_{IJ} is negative definite if and only if S_{JI} is negative definite, in Theorem 1 we need to check the definiteness of only 2^{n-1} matrices given by (3.1). Therefore, since a given matrix S_{IJ} determines a corner of the interval matrix $[B + B^T, C + C^T]$, Theorem 1 requires only the verification of the definiteness of 2^{n-1} corners of $[B + B^T, C + C^T]$. □

Corollary 1. The interval $n \times n$ matrix $[B, C]$ is Hurwitz stable if every matrix S_{IJ}, given by

$$S_{IJ} = \Lambda_{IJ} \begin{bmatrix} C_{II} & B_{IJ} \\ B_{JI} & C_{JJ} \end{bmatrix} + \begin{bmatrix} C_{II} & B_{IJ} \\ B_{JI} & C_{JJ} \end{bmatrix}^T \Lambda_{IJ},$$

is negative definite, where $\Lambda = diag[\lambda_1, \cdots, \lambda_n]$ with $\lambda_i > 0, 1 \leq i \leq n, \Lambda_{IJ}$ $= diag[\lambda_{i_1}, \cdots, \lambda_{i_s}, \lambda_{j_1}, \cdots, \lambda_{j_{n-s}}]$ and $C_{II}, C_{JJ}, B_{IJ}, B_{JI}$ are as defined in Theorem 1.

Corollary 2. Assume that B and C are symmetric matrices. The interval matrix $[B, C]$ is Hurwitz stable if and only if every matrix S_{IJ} given by

$$S_{IJ} = \begin{bmatrix} C_{II} & B_{IJ} \\ B_{JI} & C_{JJ} \end{bmatrix}$$

is negative definite, where I, J are specified in Section 2 and C_{II}, C_{JJ}, B_{IJ} and B_{JI} are as defined in Theorem 1. □

Remark 2. For symmetric matrices B and C, the interval matrix $[B, C]$ contains, in general, nondenumerably, many non-symmetric matrices. □

References

[1] M. Mansour, "Robust Stability of Interval Matrices", *Proceedings of the 28th IEEE Conference on Decision and Control*, Tampa, Florida, December 1989, pp. 46-51.

[2] M. Mansour, "Simplified Sufficient Condition for the Asymptotic Stability of Interval Matrices", *International Journal of Control*, Vol. 50, No. 1, 1989, pp. 443-444.

Robust Stability and Performance Analysis of Systems with H_∞-norm Bounded Structured Uncertainties

Z.Shao and M.E.Sawan
Department of Electrical Engineering
The Wichita State University

1. Introduction and Problem Statement

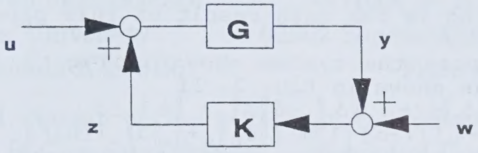


Fig.1 The normalized H_∞ problem

Consider the feedback interconnection in Fig. 1, where the system plant G and the controller K are the functions of frequency ω, with dimension n×n, and G is a rational matrix transfer function. **u**, **w** are input vectors, and **y**, **z** are output vectors, all with dimension n.

Let $\mathbf{x}=[\mathbf{u}^T,\mathbf{w}^T]^T$ and $\mathbf{v}=[\mathbf{y}^T,\mathbf{z}^T]^T$

Then the closed-loop transfer function from **x** to **v** is

$$H(G,K) = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \qquad (1)$$

where H(G,K) and $T_{ij}(i,j=1,2)$ are matrices with dimensions 2n×2n and n×n, respectively, and

$T_{11}=(I-GK)^{-1}G$; $T_{12}=(I-GK)^{-1}GK$

$T_{21}=K(I-GK)^{-1}G$; $T_{22}=K(I-GK)^{-1}$

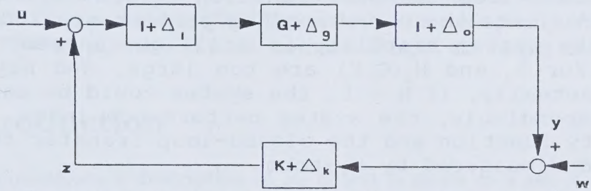


Fig. 2 Uncertainty types covered by the normalized H_∞ problem

Now, suppose system (1) has structured uncertainties as shown in Fig. 2, where Δ_i, Δ_g, Δ_o and Δ_k correspond to "input multiplicative" uncertainty, additive uncertainty on the plant G, "output multiplicative" uncertainty , and additive uncertainty on the controller K, respectively. The previous issues only consider the system with single structured uncertainty, then the motivation is arosen to study the case where the system may have multi-structured uncertainties, i.e, the system may have all uncertainty types covered in Fig. 2 at the same time. The significance to study this problem is evident. A further problem is, some performance index achieved in system (1) may not be maintained in system (2), then, can we give the estimation or upper bounds for the corresponding performance index of system (2) if the system stability is maintained? This problem is of great practical significance because structured uncertainty or perturbation is inevitable in real control systems. However, this problem has been negleted by the previous work.

2. Main Results

It is easy to show that if H(G,K) defined in (1) is internally stable, then the sensitivity function of the system defined by $S = (I-GK)^{-1}$ is also stable, because $S \equiv I+T_{12}$. So we can assume that S and H(G,K) satisfy the following inequalities: $\|S\|_\infty \le \eta$ and $\|H(G,K)\|_\infty \le \gamma$, where η and γ are positive constants.

The following is the main result of this paper.

Theorem 1. Suppose the system shown in Fig. 1 has structured uncertainties as shown in Fig. 2. If

i) $\|\Delta_i\|_\infty \le \alpha_i$; $\|\Delta_g\|_\infty \le \alpha_g$; $\|\Delta_k\|_\infty \le \beta\|K\|_\infty$; $\|\Delta_o\|_\infty \le \alpha_o$;

iv) $\alpha_o\|T_{11}\|_\infty + (1+\alpha_o)(1+\alpha_i)(1+\beta)(\|T_{11}\|_\infty + \alpha_g\|S\|_\infty) \le h\|K\|_\infty^{-1} + \|T_{11}\|_\infty$

where $\alpha_i, \alpha_g, \alpha_o, \beta$ and h are non-negative constants, $0 \le \alpha_o, h < 1$. Then

a) The sensitivity function $S_\Delta = (I-(I+\Delta_o)(G+\Delta_g)(I+\Delta_i)(K+\Delta_k))^{-1}$ remains stable and $\|S_\Delta\|_\infty \le \eta(1+\alpha_o)/[(1-\alpha_o)(1-h)]$;

b) The closed-loop transfer function $H_\Delta(G,K)$ remains stable and $\|H_\Delta(G,K)\|_\infty \le \gamma + \varphi/(1-h)$, where $\varphi = (1+\beta)h + (h+\beta)\|T_{11}\|_\infty\|K\|_\infty +$
$\max\{h(\|T_{11}\|_\infty + \|K\|_\infty^{-1}),\ \eta(1+\alpha_o)(1+\beta)\|K\|_\infty/(1-\alpha_o)\}$.

3. Conclusion

A sufficient condition is derived such that the system with multi-structured uncertainties is robustly stable. Based on this, the upper bounds measured by H_∞-norm for the sensitivity function and the closed-loop transfer function of the system are discussed. If h=0, that is, the system does not have any perturbation, the upper bounds for the sensitivity function and the closed-loop transfer function of the system derived in theorem 1 are the same as those of the nominal system. If h is small enough, then the upper bounds for the sensitivity function S_Δ and the closed-loop transfer function $H_\Delta(G,K)$ are accurate enough to approximate the corresponding performance index. As h tends to 1, the system stability is still guaranteed, but the upper bounds for S_Δ and $H_\Delta(G,K)$ are too large, and may not be significant. Actually, if $h \to 1$, the system could be marginally stable, correspondingly, the system performance index, such as the sensitivity function and the closed-loop transfer function, may be unbounded measured by H_∞-norm.

References

Daily R.Lane(1990). Lecture Notes for the Workshop on Robust Control. *American Control Conference,* San Diego, CA.

Francis B. (1987). *A course in H_∞ control theory*. New York: Springer-Velay.

Glover K. and J.C. Doyle(1988). State space formulae for all stabilizing controllers that satisfy an H_∞ norm bound and relations to risk sensitivity. *Syst. Control Lett.*, **11**, 167-172.

Kimura H. (1984). Robust stabilization for a class of transfer functions. *IEEE Trans. Aut. Control*, **AC-29**, 788-793.

Maciejouski J.M. (1989). *Multivariable feedback design.* Addision-Wesley Publishing Company.

Mustafa D. and K. Glover(1991). Controller reduction by H_∞-balanced truncation. *IEEE Trans. Aut. Control*, **36**, 668-682.

Construction of d-Dimensional Hyperoctrees on a Hypercube Multiprocessor

F. Dehne*
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6.

A. Fabri†
INRIA — B.P.93 —
06902 Sophia-Antipolis cedex,
France.

M. Nassar
Jodrey School of Computer Science
Acadia University
Wolfville, Canada B0P 1X0.

A. Rau-Chaplin*
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6.

R. Valiveti*
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6.

Abstract

We present a parallel algorithm for the construction of a d-dimensional hyperoctree from a set of n $(d-1)$-dimensional hyperoctrees, representing adjacent "slices". On a p-processor SIMD hypercube the time complexity of our algorithm is $O(\frac{m \log n}{p} \log p)$, where m is the maximum of input and output size.

1 Introduction

In many applications a description of an object is given as a set of cross sections, e.g. in medical diagnosis planar cross sections are obtained via tomography systems. In this paper we consider the problem of computing the 3-dimensional representation of an object given as a set of cross sections. There are two primary versions of this problem which can be considered depending on what is known about the input. If the cross sections are known to be widely spaced or we have no spacing information at all, i.e. the information about the 3-dimensional object consists only of planar contours extracted from the cross sectional images, then we have an interpolation problem, see for example [Bo88]. Alternatively, if the cross sections are known to lie close together, i.e. they can be considered as interpolations of the volume lying between themselves and the following

---

*Research partially supported by the Natural Sciences and Engineering Research Council of Canada.

†Research partially supported by the ESPRIT Basic Research Actions Nr. 3075 (ALCOM) and Nr. 7141 (ALCOM II).

cross section, then we have a merging problem. It is this later version of the problem that is considered in this paper.

The representation of 2 and 3 dimensional objects has been extensively studied ([R80, M88, S89]). One commonly used representation scheme, based on recursive subdivision, is the quadtree and its many variants. For an overview and bibliography on quadtrees and applications we refer to [S84]. Since in this paper we will be interested in d-dimensional representations, where d may be any constant, we will use an octree variant described in [YS83], called hyperoctrees.

Quadtrees and their higher dimensional extension are efficient spatial data structures widely used in image processing, solid modeling and many other application domains. A large number of algorithms for the construction and manipulation of these spatial data structures have been developed. Since these applications are typically data intensive, the application of parallelism to such a fundamental data structure is of both theoretical and practical interest. Recently, researchers have therefore started to consider algorithms for parallel models of computation. While some papers ([MCI86, ML86]) consider parallel architectures designed (or reconfigured) particularly for quadtree manipulation, others consider general purpose architectures, as mesh-connected computers ([HR89]) and hypercubes ([DFR91]). Many of the construction and manipulation algorithms given in these papers can be easily adapted to work for d-dimensional hyperoctrees. In particular the construction algorithms given in [DFR91] can be adapted to support the construction of hyperoctrees from both multidimensional binary images and chain codes.

In this paper we describe an algorithm for the more complex task of constructing d-dimensional hyperoctrees from sets of $(d-1)$-dimensional hyperoctrees. Given n slices of width one, the basic algorithm consists of $\log n$ phases. In each phase pairs of adjacent slices are joined to form slices of twice the width. In all phases the slices are represented by "hybrid trees", a data structure in between $d-1$ and d-dimensional hyperoctrees (to be defined later). The sequential algorithm has a running time of $O(m \log n)$ [YS83], where m is the maximum of input and output size. The time complexity of our parallel algorithm for a SIMD hypercube with p processors is $O(\frac{m \log n}{p} \log p)$. Our parallel algorithm follows the basic merging strategy of the sequential algorithm given in [YS83]. The main contribution of this paper is to solve the non trivial problem of merging "hybrid trees" in parallel.

The remainder is organized as follows. In the next section we describe our model of computation and some basic operations. In Section 3 we recall the definition of hyperoctrees and give a formal definition of a new data structure called hybrid trees. Section 4 briefly recalls the sequential algorithm and is followed by a section describing a merging algorithm for hybrid trees. Section 6 describes in detail our parallel hypercube algorithm.

2 The Model of Computation

In this section, we present two abstract models of a SIMD hypercube and some basic algorithms for this type of architecture.

A *SIMD hypercube* of dimension d consists of $p = 2^d$ processors which are indexed 0 through $2^d - 1$. Two processors are connected along dimension i, if and only if the binary representation of their indices differ in exactly the i^{th} bit. The processors are synchronized and may be enabled or disabled to execute a common instruction. Each processor has some local memory. Note that we neither assume a constant amount of memory per processor (as is done by exclusively fine-grained algorithms) nor a fixed non-constant amount of memory (as is assumed by exclusively coarse-grained algorithms). Our algorithms are suitable for implementation on either fine-grained or coarse-grained SIMD hypercubes and therefore could be implemented on machines ranging from Intel's iPSC860 [In] (using additional synchronization) to Thinking Machines Corporation's CM2 [St87].

In the first model, arithmetic operations on each processor and communication between processors which are adjacent along a fixed dimension are elementary instructions and take time $O(1)$.

In this paper we will use many *basic vector operations* such as parallel prefix, monotonic routing and bitonic merge [NS81, B68, RS90]. The parallel prefix sum of a vector V is the vector W, with $W[k] := \sum_{i=0}^{k} V[i]$, $0 \leq k < p$. Instead of summing we can perform any binary associative operation, e.g. copying. A further generalization is the segmented parallel prefix. The vector is split up in segments and the parallel prefix starts at the beginning of each segment. In this paper we also use an operation called segmented broadcast, that is in each segment all elements are replaced by the first element in the segment.

Monotonic routing refers to the following operation. Given a data vector V, a destination vector D and a Boolean vector selecting some elements of V, the selected data elements $V[i]$ are moved to $V[D[i]]$ under the condition that $D[i] < D[j]$ for all selected elements $0 \leq i < j < p$. That is, the selected data elements remain in the same order.

The bitonic merge algorithm transforms a bitonic sequence (the keys are first increasing, then decreasing) into a sorted sequence. This operation can be used to merge two sorted sequences.

All these vector operations are presented for vectors of length p and they take time $O(\log p)$. They can easily be generalized for vectors of length n, $n \geq p$, and take then time $O(\frac{n}{p} \log p)$.

The second model under consideration is the *pipelined hypercube.* Arithmetic operations again take time $O(1)$, but processors can communicate with all adjacent processors in time $O(1)$. Thus we can pipeline the above basic vector operations and get $O(\frac{n}{p} + \log p)$, $n \geq p$, as time bounds. The CM2 from Thinking Machines Corporation is an example of such a machine [BLM*91].

3 Hyperoctrees and Hybrid Trees

Hyperoctrees [YS83] are a generalization of the well known quadtrees. The d-dimensional hyperoctree, or for short 2^d-tree, is a hierarchical data structure for the representation of a d-dimensional discrete grid of sidelength n, where n is assumed to be a power of 2. The total grid is split in 2^d subgrids of half the sidelength. The subgrids are split recursively and recursion stops when a subgrid of uniform colour is reached. Nodes are either grey, black or white, representing nonuniform, black or white subgrids. The level of a node v is defined recursively. It is $\log_2 n$, if v is the root, and $\text{level}(v) := \text{level}(\text{father}(v)) - 1$ otherwise.

For our algorithm we need a further generalization of 2^d-trees, since we want to represent discrete grids where the sidelength of the side in the d-th dimension varies.

To understand why this generalization is necessary consider the following naive algorithm for combining n 2^{d-1}-trees to form a 2^d-tree.

1. Convert each of the 2^{d-1}-trees that represents a slice into an equivalent 2^d-tree.
2. Perform a union operations on all the 2^d-trees formed in the previous step.

The problem with the naive algorithm is that it can result in a considerable and unnecessary "blow up" in the amount of data that must be processed. Consider for example the situation in which all of the $(d-1)$-dimensional slices are completely black. In the first step of the naive algorithm each of the fully black $(d-1)$-dimensional slices would be broken up in n^{d-1} volume elements of sidelength 1, as the width along dimension d is only 1. Thus we would return to a representation in image space. When in the second step of the naive algorithm the union of all n slices is computed the resulting 2^d-tree is of size $O(1)$. It is exactly this unnecessary increase or "blow up" in data size that any efficient algorithm must seek to avoid.

We need to represent not only d-dimensional cubes but also cuboids with the same sidelength k in dimensions 1 through $d-1$ and sidelength $j \leq k$ in dimension d, where j is the number of already joined slices, and k and j are powers of 2.

Definition 1 A *phase i hybrid tree T^i* is a coloured tree with two kind of nodes. The nodes at levels l, $i \leq l \leq \log n$, form a 2^{d-1}-tree and represent *cuboids* of size $(2^l)^{d-1} \times 2^i$, and nodes at levels 0 through i form a forest of 2^d-trees and represent *cubes* of size $(2^l)^d$.

Notice that phase 0 or $\log n$ hybrid trees can be considered as 2^{d-1}-trees or 2^d-trees, respectively. Further note that nodes at level i are at the same time leaves of the 2^{d-1}-subtree and roots of the 2^d-subtrees. Figure 1 shows a phase $i=2$ hybrid tree for $d=2$. The nodes at level $l>2$ represent rectangles of height $2^i=4$ and length 2^l, nodes at levels $l \leq 2$ represent squares of sidelength 2^l.

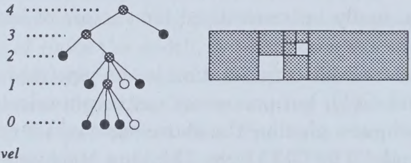


Figure 1: A hybrid tree

4 Sequential Algorithm

The above definition of hybrid trees formalizes an intermediate state of the sequential algorithm presented in [YS83]. The basic structure of their algorithm is to consider each 2^{d-1}-tree as a phase 0 hybrid tree and perform $\log n$ pairwise merge phases. More precisely, in phase $i+1$ of the merge algorithm, $0 \leq i < logn$, pairs T_1^i and T_2^i of phase i hybrid trees are merged into a phase $i+1$ hybrid tree $T_{1,2}^{i+1}$.

In the following section, we present a modified version of this merge phase, which we will use in our parallel algorithm.

5 Merging Hybrid Trees

We describe an algorithm to join two hybrid trees T_1^i and T_2^i in phase $i+1$ of the merge. For this algorithm to be efficient it must avoid performing work that will later have to be undone. In particular, the decision to break up a cuboid should be postponed as long as possible, i.e. until it is certain that the cuboid will never become part of a uniformly coloured cube. When a cuboid is broken up, it must be broken up into cuboids which are maximal and which also fit this criterion.

Before we can join the trees we overlay the 2^{d-1}-subtrees of T_1^i and T_2^i and expand them to the same shape. While expanding the trees, we consider three different cases for all corresponding nodes v_1 and v_2 at levels $l > i$ in the two trees.

Case 1: If v_1 is black or white and v_2 is grey, then the cuboid of v_1 must be broken up to the same level of resolution as the cuboid of v_2. Therefore we replace v_1 with a copy of the 2^{d-1}-subtree rooted at v_2, where leaf nodes get the colour of v_1. We do not have to copy nodes below level i, as nodes in level i already represent cubes.

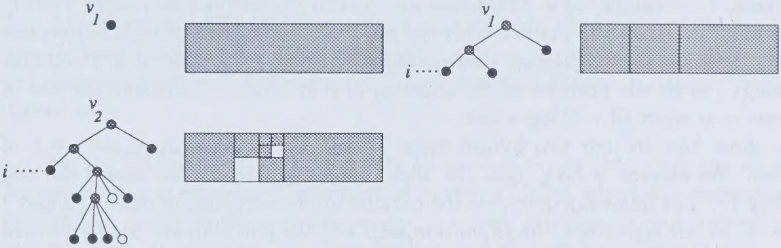


Figure 2: Case 1 for $d = 2$

Case 2: If v_1 is black and v_2 white, they must both be broken up to cubes of sidelength 2^i. We thus replace each of them with the complete 2^{d-1}-tree of height $l - i$ with leaf nodes coloured as v_1 and v_2, respectively.

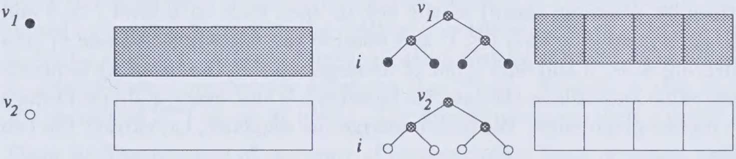


Figure 3: Case 2 for $d = 2$

Case 3: If they are both either black or white, they represent cuboids of the same length and form together a cuboid of height 2^{i+1}, and thus no expansion is necessary.

The 2^{d-1}-subtrees of the two trees are now of the same shape and corresponding nodes on levels $l \geq i + 1$ have the same colour. We use the overlayed trees to construct the 2^d-tree nodes on level $i + 1$, by joining the children of corresponding grey 2^{d-1}-tree nodes at level $i + 1$. Keeping only one copy of the expanded 2^{d-1}-trees we obtain the hybrid tree $T_{1,2}^{i+1}$.

On a sequential machine the above merge operation can be easily implemented in time $O(\max(|T_1^i| + |T_2^i|, |T_{1,2}^{i+1}|))$ by performing simultaneous tree traversals [YS83]. As the overall algorithm consists of $\log n$ phases, the input set of n $(d-1)$-dimensional hyperoctrees can be transformed into the corresponding d-dimensional hyperoctrees T in time $O(m \log n)$, with $m = \max(|T|, \sum_{j=1}^{n} |T_j|)$.

6 Parallel Algorithm

In the parallel setting, an efficient implementation of hybrid tree merge is not as straightforward. Since the sequential time complexity is $\Theta(m)$, an optimal parallel algorithm, even for the PRAM model, can achieve a time not better than $O(\frac{m}{p})$. In the following, we present an $O(\frac{m}{p}\log p)$ time hypercube algorithm.

We will use a linear representation instead of the pointer based representation of hybrid and 2^d-trees used in the sequential setting. A *linear tree* is a collection of leaf nodes, ordered according to the inorder traversal of the pointer based tree.

For each node v in the 2^{d-1}-subtree of a hybrid tree, we define the position of v, $pos(v)$, to be the inorder traversal number of its corresponding node w in the complete $(d-1)$-ary tree, $\bar{T}$, of height $\log n$. We define $cover(v)$ to be the interval $[lm(v), rm(v)]$, where $lm(v)$ and $rm(v)$ are the inorder traversal numbers (with respect to the complete tree $\bar{T}$) of the leftmost and rightmost descendant of w. For each node in a 2^d-subtree we define $anc_i(v)$ to be the position of the ancestor of v in level i. Note that the size of these numbers is at most $(d-1)\log n$ bits.

We now show how to join two hybrid trees T_1^i and T_2^i to perform phase $i+1$ of the algorithm. We assume w.l.o.g. that the slice represented by T_1^i lies above the one represented by T_2^i. The following describes the parallel implementation of cases 1, 2 and 3 from Section 3. In our algorithm, the expansion step and the join step are not performed separately.

Case 1: In this case each black or white node, v_1, in T_1^i corresponding to a grey node, v_2, in T_2^i is replaced by a tree of the shape of T_{v_2}, the 2^{d-1}-subtree of the tree T_2^i rooted at v_2. As internal nodes are not represented by the linear representation, this is equivalent to replacing v_1 by the leaves of T_{v_2}. We perform this operation in two steps.

In the *first step* we detect all black and white nodes in levels $l > i$ of T_1^i, which correspond to grey nodes of T_2^i. These black and white nodes are called *covering nodes.* We merge the trees by assigning $pos(v)$ as the key to each node in a level $l > i$ and $anc_i(v)$ as key to each node in a level $l \leq i$, and observe the following. A node v_1 in a level $l > i$ is a covering node, if and only if one of its neighbours in the resulting sequence has a different pos value and falls in the interval $cover(v_1)$. Thus, every node performs a comparison with its two neighbours. We then unmerge the sequence, i.e. extract the two original sequences.

In the *second step* we determine for each covering node v_1 the set of leaves of the subtree T_{v_2} rooted at the corresponding node v_2. We first make a copy of each covering node and store it in an adjacent position. Every node v in T_1^i or T_2^i is then assigned a key, $key(v) = (x, y)$, where x and y are the primary and secondary keys, respectively. We $key(v)$ as follows:

$$key(v) = \begin{cases} (lm(v), 0) & \text{if } v \text{ is the first copy of a covering node in } T_1^i \text{ at level } l > i, \\ (rm(v), 3) & \text{if } v \text{ is the second copy of a covering node in } T_1^i \text{ at level } l > i, \\ (pos(v), j) & \text{if } v \text{ is a noncovering node in } T_j^i,\ j{=}1{,}2, \text{ at level } l > i, \\ (anc_{i+1}(v), j) & \text{if } v \text{ is a node in } T_j^i,\ j{=}1{,}2, \text{ at level } l \leq i. \end{cases}$$

Lemma 1 *The nodes in the linear hybrid trees form a monotonic sequence with respect to key(v).*

Proof: The nodes are ordered by their *pos* value. As only leaf nodes are represented, all $lm(v)$ and $rm(v)$ values are unique and no *pos* value of another node falls in the interval *cover*(v). Noncovering nodes at level $l > i$ do not change their key. Let $(u_1 \ldots u_k)$ be a maximal subsequence of nodes of the linear tree T_j, which are at level $l \leq i$ and receive the same key. Let w be their ancestor in level $i+1$, then *pos*$(u_1) <$*pos*$(w) <$*pos*(u_k). □

Thus, we can join the two trees using a stable bitonic merge. The following lemma states some observations on the resulting sequence S. We omit the proof of the statements as they are a direct consequence of the choice of *key*(v).

Lemma 2 *The sequence S has the following properties:*
(a) *The first copy of a covering node appears immediately to the left of the first node in the sequence of covered nodes, the second copy immediately to the right of the last node in the sequence of covered nodes.*
(b) *The 2^d-trees which become children of a node in level $i+1$ of the resulting hybrid tree $T_{1,2}^{i+1}$ are in the order required for $T_{1,2}^{i+1}$.*
(c) *Corresponding black or white 2^{d-1}-tree nodes of T_1^i and T_2^i, respectively, are neighbours in S.*

We use Lemma 2(a) to compute for all covered nodes their covering node. We do so by performing a segmented broadcast of covering nodes, where the segments are defined by the two copies of the covering nodes. As *cover* intervals do not overlap, any node is covered by at most one node. Then both copies of covering nodes are removed from S.

We now create the copies of the leaves of T_{v_2}, the 2^{d-1}-subtree rooted at v_2, for the covering node v_1. Each covered subsequence $\hat{S}$ of 2^d-tree nodes with the same key forms the set of leaves of a subtree rooted in level $i+1$. We thus insert 2^{d-1} copies of the covering node before or behind $\hat{S}$, depending on, whether the covering node stems from T_1^i or T_2^i.

Case 2: In this step we expand corresponding nodes v_1 and v_2 where one of the nodes is black the other node white. According to lemma 2(c) corresponding nodes can be determined by a comparison with the neighbour in the sequence S. Let the two corresponding nodes be at level l. Instead of replacing them first by two 2^{d-1}-trees which are then joined in a second step, we directly construct the 2^d-tree. To achieve this we replace them by 2^{i-l} groups of 2^{d-1} nodes at level i. The groups alternate their colours as v_1 and v_2 do.

Case 3: The removal of one copy of corresponding black or white nodes completes the joining of two 2^{d-1}-trees and yields the tree $T_{1,2}^{i+1}$.

This concludes our discussion of a phase $i+1$ merge of two trees T_1^i and T_2^i. As we start with n hyperoctrees $T_1, \ldots, T_n$, we need $\log n$ such phases to obtain the final 2^d-tree, T. The time we need to perform one phase depends on the input size as well as on the output size. This can be illustrated with the following examples. If all slices are black, we start with n single node trees and end up with one node representing a black block. On the other hand, if slices are alternating black and white, we start with n single node trees and end up with one tree of size n^d. As the algorithm only constructs nodes which either appear in the final output tree T or which are further refined, we can estimate the data size with $m = \max(|T|, \sum_{i=1}^{n} |T_i|)$.

The processor load may change in each phase and the insertion and deletion of nodes is in fact a reallocation of processors. This can be performed with parallel prefix and monotonic routing operations. Since the nodes always remain in the same order it requires time $O(\frac{m}{p} \log p)$.

In a preprocessing step we compute the keys *pos* in time $O(\frac{m}{p}\log m)$ using local operations. For each node v we start with the position of the root and follow the path from the root to v. It takes time $O(1)$ to compute the position of a child of node v, given the position of v. The height of the tree is $O(\log m) = O(d \log n) = O(\log n)$.

Given *pos*, the value of *lm* and *rm* can be computed in time $O(1)$ using closed formulas as the tree is a complete tree.

In each phase we have to compute keys for nodes which were either inserted or which changed from the 2^{d-1}-subtree to a 2^d-subtree. Nodes which are inserted obtain their keys from the nodes which caused their insertion. Given anc_i the value of anc_{i+1} can be computed in time $O(1)$.

This results in the following theorem.

Theorem 1 *The construction of a d-dimensional hyperoctree from a set of* n $(d-1)$*-dimensional hyperoctrees on a hypercube with p processors takes time* $O(\frac{m \log n}{p} \log p)$*, where m is the maximum of input and output size.*

We immediately get the following corollary for pipelined hypercubes

Corollary 1 *The construction of a d-dimensional hyperoctree from a set of* n $(d-1)$*-dimensional hyperoctrees takes time* $O(\frac{m \log n}{p} + \log p \log n)$ *on a p processors pipelined hypercube, where m is the maximum of input and output size.*

7 Conclusion

In this paper we have presented a parallel algorithm for the construction of a d-dimensional hyperoctree from a set of n $(d-1)$-dimensional hyperoctrees, representing adjacent "slices". On a p-processor SIMD hypercube the time complexity of our algorithm is $O(\frac{m \log n}{p} \log p)$, where m is the maximum of input and output size. This parallel algorithm represents, to our knowledge, the first parallel algorithm given for the construction of octree based data structures from anything but binary images or chain-codes.

References

[B68] K.E. Batcher. *Sorting networks and their applications.* Proc. AFIPS Spring Joint Computer Conference, pp. 307–314, 1968.

[BRW88] S. K. Bhaskar, A. Rosenfeld, and A. Y. Wu. *Parallel processing of regions represented by linear quadtrees.* Computer Vision, Graphics, and Image Processing, Vol. 42, pp. 371–380, 1988.

[Bo88] J.D. Boissonnat. *Shape Reconstruction from Planar Cross Sections.* Computer Vision, Graphics and Image Processing 44, pp. 1–29, 1988.

[DFR91] F. Dehne, A. Ferreira, and A. Rau-Chaplin. *Parallel processing of pointer based quadtrees.* Proc. International Conference on Parallel Processing, 1991.

[HR89] Y. Hung and A. Rosenfeld. *Parallel processing of linear quadtrees on a mesh-connected computer.* Journal of Parallel and Distributed Computing, Vol. 7, pp. 1–27, 1989.

[In] Intel Scientific Computers *iPSC System Overview.*

[M88] M. Mäntylä. *An Introduction to Solid Modeling.* Computer Science Press, 1988.

[MCI86] M. Martin, D. M. Chiarulli, and S. S. Iyengar. *Parallel processing of quadtrees on a horizontally reconfigurable architecture computing system.* Proc. International Conference on Parallel Processing, pp. 895–902, 1986.

[ML86] G.-G. Mei and W. Liu. *Parallel processing for quadtree problems.* Proc. International Conference on Parallel Processing, pp. 452–454, 1986.

[NS81] D. Nassimi and S. Sahni. *Data broadcasting in SIMD computers.* IEEE Trans. on Computers 30:2, pp. 101–106, 1981.

[RS90] S. Ranka and S. Sahni. *Hypercube Algorithms: with Applications to Image Processing and Pattern Recognition* Bilkent University Lecture Series, Springer-Verlag, 1990.

[R80] A. G. Requicha. *Representation for Rigid Solids: Theory, Methods and Systems.* ACM Computing Surveys, Vol. 12, No. 4, 1980.

[S84] H. Samet. *The quadtree and related hierarchical data structures.* Computing Surveys, Vol. 16, No. 2, pp. 187–260, 1984.

[S89] H. Samet. *Applications of Spatial Data Structures.* Addison-Wesley, 1989.

[St87] C. Stanfill. *Communications Architecture in the Connection Machine System.* Thinking Machines Corporation — Technical Report HA87-3, 1987.

[BLM*91] G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, M. Zagha. *A Comparison of Sorting Algorithms for the Connection Machine CM-2.* ACM Symposium on parallel algorithms and architectues, 1991.

[YS83] M. Yau and S. N. Srihari. *A Hierarchical Data Structure for Multidimensional Digital Images.* Comm. of the ACM, No. 1, Vol. 26, July 1983.

Optimal Mesh Algorithms for the Voronoi Diagram of Line Segments, Visibility Graphs and Motion Planning in the Plane

Sanguthevar Rajasekaran Suneeta Ramaswami*

Department of Computer and Information Science,
University of Pennsylvania, Philadelphia, PA 19104

Abstract The motion planning problem for an object with two degrees of freedom moving in the plane can be stated as follows: Given a set of polygonal obstacles in the plane, and a two-dimensional mobile object B with two degrees of freedom, determine if it is possible to move B from a start position to a final position while avoiding the obstacles. If so, plan a path for such a motion. Techniques from computational geometry have been used to develop exact algorithms for this fundamental case of motion planning. In this paper we obtain optimal mesh implementations of two different methods for planning motion in the plane. We do this by first presenting optimal mesh algorithms for some geometric problems that, in addition to being important substeps in motion planning, have numerous independent applications in computational geometry.

In particular, we first show that the *Voronoi diagram* of a set of n nonintersecting (except possibly at endpoints) *line segments* in the plane can be constructed in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh, which is optimal for the mesh. Consequently, we obtain an optimal mesh implementation of the sequential motion planning algorithm described in [15]; in other words, given a disc B and a polygonal obstacle set of size n, we can plan a path (if it exists) for the motion of B from a start position to a final position in $O(\sqrt{n})$ time on a mesh of size n. Next we show that given a set of n line segments and a point p, the set of segment endpoints that are visible from p can be computed in $O(\sqrt{n})$ mesh-optimal time on a $\sqrt{n} \times \sqrt{n}$ mesh. As a result, the *visibility graph* of a set of n line segments can be computed in $O(n)$ time on an $n \times n$ mesh. This result leads to an $O(n)$ algorithm on an $n \times n$ mesh for planning the shortest path motion between a start position and a final position for a convex object B (of constant size) moving among convex polygonal obstacles of total size n.

1 Introduction

The problem of algorithmic motion planning has received considerable attention in recent years. The automatic planning of motion for a mobile object moving amongst obstacles is a fundamentally important problem with numerous applications in computer graphics and robotics. The study of *algorithmic* techniques for planning motion, with provable worst-case performance guarantees, has been spurred by recent research that has established the mathematical depth of this problem (see [19, 20] for comprehensive surveys). In particular, the design and analysis of geometric algorithms has proved to be very useful, resulting in considerable interplay between computational geometry and algorithmic motion planning for numerous special cases.

We are interested in studying special cases of algorithmic motion planning and the related geometric problems using parallelism. For a number of special cases of motion planning, optimal or near optimal sequential algorithms have been discovered. Our research aims at obtaining optimal parallel algorithms for these problems and will be aided by the significant progress that has been made in the area of parallel algorithms for computational geometry in recent years ([1, 4, 6, 7, 8, 13, 16], for example).

In this paper we develop efficient parallel *mesh* algorithms for two different techniques of planning motion for an object with two degrees of freedom moving in the plane among polygonal obstacles. One technique, first developed by Ó'Dúnlaing and Yap [15], uses *Voronoi diagram* construction for a set of *line segments* as a subroutine and the other technique, suggested by Lozano-Pérez and Wesley [12], uses planar *visibility graph* construction.

Visibility graph construction and Voronoi diagrams are geometric problems which, in addition to being tools for motion planning, have many useful applications. Atallah *et al.* [4] give an efficient PRAM algorithm for computing visibility from a point, given an arbitrary set of nonintersecting line segments and Goodrich *et al.* give a CREW PRAM algorithm for constructing the Voronoi diagram of a set of line segments in the plane [7]. However, to our knowledge, these problems have not been solved on fixed-connection networks. In this paper, we develop efficient parallel algorithms for these geometric problems on the mesh-connected-computer and as a result, for the corresponding motion planning problems.

The *mesh-connected computer* (*mesh*) of size n is a fixed-connection network of n simple *processing*

---

*This research is partially supported by ARO Grant DAAL03-89-C-0031 including participation by the U.S. Army Human Engineering Laboratory.

elements (PEs) that are arranged in a $\sqrt{n} \times \sqrt{n}$ two-dimensional grid. Each PE is connected to its (at most) four nearest neighbors. Attractive features such as simple near-neighbor wiring and ease of scalability have made the mesh-connected computer the focus of considerable attention in parallel algorithms research. The following mesh operations, which will be used in the remainder of this paper, can be implemented in $\theta(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh[13]: perfect shuffle, perfect unshuffle, sorting, selected broadcasting, segmented prefix scan, Random Access Read (RAR), and Random Access Write (RAW). The commonly used indexing schemes on the mesh are row-major, shuffled row-major, snake-like and proximity [13]. An implicit lower bound of $\Omega(\sqrt{n})$ holds for most algorithms on the mesh, because nontrivial data movement takes $\Omega(\sqrt{n})$ steps.

In the next section we give some important definitions and a brief introduction to relevant background. In Section 3, we develop an optimal mesh algorithm for the construction of the Voronoi diagram of a set of line segments in the plane. We summarize the resulting mesh algorithm for the related motion planning technique at the end of that section. In Section 4, we give an optimal mesh algorithm for determining visibility from a point, and therefore for visibility graph construction. The resulting mesh algorithm for planning motion is also given in that section.

2 Background

The motion planning problem of interest to us can be stated in the following way: Given an initial starting position P_I, a final destination position P_F and a set of stationary obstacles whose geometry is known to object B, determine if there exists a continuous obstacle-avoiding motion for B from P_I to P_F. If one exists, construct the path for such a motion. Let n be the size of the obstacle set and let k be the number of degrees of freedom[1] (*dofs*) of the mobile object B. Every position of B can be thought of as a point in k-dimensional parametric space. Let a *free configuration* be a placement of B in which it does not intersect with any of the obstacles. Define FP to be the subset of k-dimensional space that contains all the free configurations of B. In general, FP will consist of many path-connected components. A collision-free path from P_I to P_F exists if and only if the corresponding k-dimensional configurations lie in the same connected component.

There are, in general, two kinds of strategies to solve the motion planning problem. The first general approach, which runs in time polynomial in n and doubly exponential in k, was first demonstrated by Schwartz and Sharir in [17], and has been applied to numerous special cases. The important step in this approach is to construct a *connectivity graph* that represents the connectivity information of the cells of FP. Planning motion for B then reduces to performing a graph search on the connectivity graph. The second general approach is to find a one-dimensional representation of FP (called the "skeleton" or the "road-map") such that it is possible for B to move from P_I to P_F iff it is possible to move between two corresponding points on the skeleton. This generalized approach was given by Canny [5] and runs in time polynomial in n and single exponential in k. Techniques based on the ideas of the first approach are called the *projection methods*, and those based on the second approach are called the *retraction methods.* The planar motion planning methods that use Voronoi diagrams [15] and visibility graphs [12] are instances of the retraction and projection method, respectively.

2.1 Notation and Definitions

Voronoi Diagram of a Set of Line Segments in the Plane Let S be a set of non-intersecting closed line segments in the plane. Following the convention in [10, 22], we will consider each segment $s \in S$ to be composed of three distinct objects: the two endpoints of s and the open line segment bounded by those endpoints. Following [7, 10], we now establish some basic definitions. The Euclidean distance between two points p and q is denoted by $d(p,\ q)$. The *projection* of a point q on to a closed line segment s with endpoints a and b, denoted $proj(q,\ s)$, is defined as follows: Let p be the intersection point of the straight line containing s (call this line $\overleftrightarrow{s}$), and the line going through q that is perpendicular to $\overleftrightarrow{s}$. If p belongs to s, then $proj(q,\ s) = p$. If not, then $proj(q,\ s) = a$ if $d(q,\ a) < d(q,\ b)$ and $proj(q,\ s) = b$, otherwise. The *distance* of a point q from a closed line segment s is nothing but $d(q,\ proj(q,\ s))$. By an abuse of notation, we denote this distance as $d(q,\ s)$. Let s_1 and s_2 be two objects in S. The *bisector* of s_1 and s_2, $B(s_1,\ s_2)$, is the locus of all points q that are equidistant from s_1 and s_2 i.e. $d(q,\ s_1) = d(q,\ s_2)$. Since the objects in S are either points or open line segments, the bisectors will either be parts of lines or parabolas.

Definition 2.1 *[10] The* Voronoi region, $Vor(e)$, *associated with an object e in S is the locus of all points that are closer to e than to any other object in S i.e.* $Vor(e) = \{p \mid d(p,e) \leq d(p,e')$ *for all* $e' \in S\}$. *The* Voronoi diagram *of* S, $Vor(S)$, *is the union of the Voronoi regions* $Vor(e), e \in S$. *The boundary edges of the Voronoi regions are called* Voronoi edges, *and the vertices of the diagram,* Voronoi vertices.

---

[1] The degrees of freedom of an object can be defined as the number of parameters that need to be specified in order to completely determine the position of the object.

Theorem 2.2 (Lee et al. [10]) *Given a set S of n nonintersecting closed line segments in the plane, the number of Voronoi regions, Voronoi edges, and Voronoi vertices of $Vor(S)$ are all $O(n)$.*

The above theorem states a very important property of $Vor(S)$. Sequential algorithms for the construction of the Voronoi diagram of a set of line segments are given by Kirkpatrick [9], Lee and Drysdale [10], and Yap [22]. The algorithms in [9, 22] run in $O(n \log n)$ time, which is optimal since a lower bound of $\Omega(n \log n)$ is known for this problem[18]. The run-time of the algorithm in [10] is $O(n \log^2 n)$. We will repeatedly refer to Yap's algorithm in the coming sections, since it lends itself to efficient parallelization, whereas the other two techniques do not. Goodrich *et al.* [7] give a CREW PRAM algorithm for Voronoi diagram construction that uses n processors and runs in $O(\log^2 n)$ time.

Visibility Graphs Given a set S of n line segments in the plane, its *visibility graph* G_S is the undirected graph which has a node for every endpoint of the segments in S, and in which there is an edge between two nodes if and only if they are visible to each other, assuming the line segments are opaque. Welzl [21] and Asano *et al.* [2] give sequential algorithms for constructing the visibility graph of a set S of line segments with $|S| = n$ that run in $O(n^2)$ time. Unfortunately, neither of these two sequential techniques lends itself to efficient parallelization.

The problem of computing visibility from a point, i.e. identifying those vertices of S that are visible from some specified point p, has a lower bound of $\Omega(n \log n)$. This lower bound can be established by showing a straightforward reduction from sorting, assuming that we want the output in sorted order about p (by polar angle with respect to some fixed axis through p). Atallah *et. al.* [4] give an optimal CREW PRAM algorithm for computing visibility from a point that runs in $O(\log n)$ time using n processors. The visibility graph can thus be constructed by repeating this algorithm for each of the endpoints of S, which takes $O(\log n)$ time with n^2 processors. When the input is a disjoint set of polygons, the visibility graph is constructed by using the polygon edges as the input set S.

3 Mesh Algorithms for the Voronoi Diagram of a Set of Line Segments and the Related Motion Planning Problem

As we mentioned in Section 1, the Voronoi diagram turns out to be a useful tool in motion planning [14, 15]. We now describe a mesh-optimal algorithm for the construction of the Voronoi diagram of a set of line segments in the plane. The resulting mesh implementation of the motion planning algorithm by Ó'Dúnlaing and Yap [15] is given in the last part of this section.

3.1 Voronoi Diagram of a Set of Line Segments in the Plane

In this section, we develop a parallel algorithm for constructing the Voronoi diagram of a set of N line segments in the plane on a $\sqrt{n} \times \sqrt{n}$ mesh ($n = 2N$) that runs in $O(\sqrt{n})$ time, which is optimal for the mesh. We would like to point out that there is an optimal $O(\sqrt{n})$ time parallel algorithm for the Voronoi diagram of a set of n *points* in the plane, on a mesh with as many PEs (Jeong and Lee [8]), but none, to our knowledge, for line segments.

The general idea behind the sequential algorithms for the construction of $Vor(S)$ (S is the input set of line segments) is as follows: S is divided into sets of equal size, S_1 and S_2. $Vor(S_1)$ and $Vor(S_2)$ are then recursively computed. In order to merge these two Voronoi diagrams to form the final diagram $Vor(S)$, we need to construct the *contour* between S_1 and S_2. The contour is the locus of all points in the plane that are equidistant from S_1 and S_2. We discard that part of the diagram of $Vor(S_1)$ ($Vor(S_2)$) that lies to the right (left) of the contour. The remaining edges of the two diagrams, and the contour edges give us the final Voronoi diagram $Vor(S)$. This is the motivation behind the sequential approaches used by [9, 10, 22].

Thus, the construction of the contour is the single most important step in the merge phase of the divide-and-conquer algorithm for Voronoi diagram construction. For the case of a set of points in the plane, it is easy to find S_1 and S_2 so that we have the nice property that there is exactly one contour to be constructed, and this contour is monotone with respect to the y axis. For the Voronoi diagram of line segments, however, it is much more complicated to ensure that this property of the contour holds. The CREW PRAM algorithm by Goodrich *et al.* makes use of data structures that are of size $O(n \log n)$. We cannot make use of such data structures if we assume constant storage per PE on a mesh-connected-computer of size n. In addition, their method performs numerous pointer manipulations, which are difficult to implement on the mesh. Our mesh algorithms avoids these pitfalls. Before we proceed, we state two results that are of relevance to Voronoi diagram construction on the mesh.

Lemma 3.1 *Given a linearly ordered set of elements L and a set of elements E such that each $e \in E$ lies between exactly two elements of L (call these e^a and e^b), and $|L| + |E| = n$. The problem of finding e^a and e^b for every $e \in E$ can be solved in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh. Call this Algorithm* SIMULTSRCH.

Proof: Omitted. □

Lemma 3.2 (Jeong and Lee [8]) *Given an arbitrary set of segments S in the plane and a set of points P such that $|S| + |P| = n$. Let p^a (p^b) be the segment from S that lies immediately above (below) p. The problem of finding p^a and p^b for every point $p \in P$ (also known as the* MultiLocation *problem) can be solved in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh. Call this Algorithm* MultiLoc.

Let $S = \{s_0, s_1, \ldots, s_{N-1}\}$ be the input set of line segments that do not intersect (except possibly at endpoints). Let v_{2i} and v_{2i+1} be the two endpoints of segment s_i, such that $x(v_{2i}) < x(v_{2i+1})$. Each segment s of S is actually represented as three elements: the two endpoints and the open line segment. Let $E = \{p_0, p_1, \ldots, p_{n-1}\}$ be the ordered set consisting of these endpoints sorted according to their x-coordinates (each p_j is some v_i and $n = 2N$; E could be a multiset). The mesh algorithm for constructing $Vor(S)$ will be a divide-and-conquer algorithm, and so we will assume shuffled row-major indexing on the mesh. Suppose a vertical line is drawn through each point in S. The vertical strip of region between any two such (not necessarily adjacent) vertical lines is called a *slab*. Consider the set of segments that span a slab U. The region of U that is enclosed between two such *consecutive* spanning segments is called a *quad* of U. A quad is said to be an *active quad* if it contains an endpoint of S in its interior. The subset of E in the interior of U will be referred to as E_U (thus, endpoints lying on the vertical boundaries of U do not count). The set of segments obtained by restricting S to the slab U will be called S_U i.e. $S_U = \{s \cap U \mid s \in S \text{ and } s \cap U \neq \emptyset\}$. Yap's sequential algorithm is a divide-and-conquer algorithm that computes the Voronoi diagram for the segments in each slab. However, a naive implementation of this strategy would take $O(n^2)$ time in the worst case. Yap overcomes this by computing, for every slab U, the Voronoi diagram for only those segments of S_U that belong to some active quad of U.

Let U be the slab obtained by merging the adjacent slabs U_1 and U_2. The merge step computes the Voronoi diagram in all the active quads of U; this is done by using, with some additional computation, the recursively computed Voronoi diagrams of the active quads of U_1 and U_2 to construct the contour. Thus, the most important step in the merge procedure is to compute efficiently $Vor(S_U \cap Q)$ for every active quad Q in U. Following [7], we let $VorSet(S_U)$ represent the set containing the Voronoi diagrams of all the active quads Q of U i.e. $VorSet(S_U) = \{Vor(S_U \cap Q) \mid Q \text{ is an active quad of } U\}$. At the topmost level of recursion, the entire plane is the slab U, and the algorithm computes $Vor(S)$, since $VorSet(S_U)$ is nothing but $Vor(S)$.

Initially each PE P_i, $0 \leq i \leq n-1$ has a packet that contains v_i, $s_{\lfloor i/2 \rfloor}$ and $v_{i+(-1)^i}$ and v_i is used as the key for processor P_i's information.

Preprocessing: In this step, (a) first we sort the packets according to the x-coordinate of the key. Notice that now the arrangement of the keys of the packets is as in the ordered set E. (b) Next, we run Algorithm MultiLoc (refer Lemma 3.2), using S and E as the set of segments and points, respectively. The packet in PE P_i now has p_i, ${p_i}^a$ and ${p_i}^b$. It will become clear later on that this preprocessing step is necessary in order to determine active quads. Clearly, (a) and (b) take $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh.

Base Step: The base step is executed when there is exactly one point in the interior of the slab. This point will be p_i, for odd i, $1 \leq i \leq n-1$. The active quad to which p_i belongs (obviously, it is the only active quad in p_i's slab) is given by the spanning segments ${p_i}^a$ and ${p_i}^b$. Clearly, the Voronoi diagram of this quad can be computed in constant time. Hence the base step takes constant time.

Merging: Let U_l and U_r be two adjacent slabs, and let $|E_{U_l}| = |E_{U_r}| = k$ (i.e. each slab has k endpoints in its interior). Suppose that $VorSet(S_{U_l})$ and $VorSet(S_{U_r})$ have been recursively computed in two adjacent sub-blocks of the mesh, where each sub-block is of size $\sqrt{k+1} \times \sqrt{k+1}$. Let the left sub-block be called M_l and the right sub-block M_r. We will show that we can perform the merge in $O(\sqrt{k})$ time, using $O(k)$ PEs.

The information that is necessary for the merge procedure is available in M_l in the following manner. (1) *Active Quads of U_l :* The active quads in U_l have a sorted order defined on them in the natural way. Let A_l be the number of active quads in U_l ($A_l \leq k$); let these be $Q_{l1}, Q_{l2}, \ldots, Q_{lA_l}$ in sorted order (from top to bottom, say). See Figure 1 for an example. Let the number of endpoints in these active quads be $k_{l1}, k_{l2}, \ldots, k_{lA_l}$, respectively. Note that $k_{l1} + k_{l2} + \ldots + k_{lA_l} = k$. In M_l, the endpoints in Q_{l1} are in the first k_{l1} processors, the endpoints in Q_{l2} are in the next k_{l2} processors and so on. We will call this the *active-quad-wise ordering* of the endpoints of E_{U_l}. Each endpoint in Q_{li} will specify its quad by the upper and lower bounding segments of Q_{li}. (2) *Voronoi Edges of $VorSet(S_{U_l})$:* Because of the quad-wise computation of the Voronoi diagram, the Voronoi edges of $VorSet(S_{U_l})$ are stored in a quad-wise manner. In other words, in M_l, we will first have the Voronoi edges of $Vor(S_{U_l} \cap Q_{l1})$, followed by the edges of $Vor(S_{U_l} \cap Q_{l2})$, and so on. We would like to note that the PEs of M_l that store active quad Q_{li} suffice to store the complete diagram $Vor(S_{U_l} \cap Q_{li})$, with just a constant number of Voronoi edges per PE (this fact follows directly from a lemma by Yap in [22]).

Let A_r be the number of active quads of U_r, and let k_{ri} be the number of points in the i-th (in the sorted order) active quad Q_{ri}, $1 \leq i \leq A_r$ (see Figure 1). The information about the active quads of U_r and the Voronoi edges of $VorSet(S_{U_r})$ are available in M_r in a similar and analogous way.

For the sake of brevity, we will give a very general description of the merge step on the mesh without going into the details.

Summary of the Merge Step on the Mesh
The merge part of this divide-and-conquer algorithm consists of three important substeps [22]: the determination of the active quads of U, the *vertical merge*, and the *horizontal merge*.

(1) **Determination of the active quads of U :** In this step we compute the active quads of U by using the information about the active quads of U_l and U_r available in M_l and M_r, respectively. This is done by merging the endpoints in M_l with the endpoints in M_r according to the upper bounding segment of the quad that they belong to (some Q_{li} or Q_{ri}). This merge can be done by performing the standard shuffle-exchange step. This step ensures that all the points in E_U lie in $M_l \cup M_r$ in the correct active-quad-wise ordering. An appropriate selected broadcasting step can now update, for every endpoint in E_U, the upper and lower bounding segments of the active quad of U that it lies in. This step takes $O(\sqrt{k})$ time on the mesh $M_l \cup M_r$ (which has $2k + 2$ PEs).

Note: Consider an active quad Q from the slab U. Let Q_l (Q_r) represent the part of Q that lies in the left (right) slab U_l (U_r). In other words, $Q_l = Q \cap U_l$ and $Q_r = Q \cap U_r$. Observe that Q_l (Q_r) is the union of a contiguous set of quads of slab U_l (U_r). Some of these quads may be active and some or all of them may not be (see Figure 1 for an example). We will call these quads (whether active or not) the Q_l-quads (Q_r-quads). In order to find the Voronoi diagram of Q, $Vor(S_U \cap Q)$, we need to "merge" the Voronoi diagrams of all the Q_l-quads and the Q_r-quads in the appropriate way. This merging is achieved by first doing a *vertical merge*, followed by a *horizontal merge*.

(2) **The vertical merge:** In this step we find, for every active quad Q of U, the Voronoi diagram of $S_{U_l} \cap Q_l$, called the Q_l-diagram and of $S_{U_r} \cap Q_r$, called the Q_r-diagram. Notice that the Voronoi diagrams of the non-empty Q_l-quads (Q_r-quads) have already been recursively computed. The Voronoi diagram of an empty Q_l-quad (Q_r-quad) is of constant size and thus can be computed in constant time. Hence, determining the empty quads is the important step.

Consider an empty Q_l-quad; call it Q'. On the mesh $M_l \cup M_r$, we arrange the upper and lower bounding segments of Q' in such a way that there are no endpoints of E_{U_l} between the two processors that hold these segments. In addition, we arrange all the Q_l-quads, whether empty or active, in the correct sorted order on the mesh (it is clear that such a sorted order on all Q_l-quads is well-defined). Similarly for the Q_r-quads. By defining an appropriate ordering on all the endpoints of E_U, we can sort them into the arrangement described above. We will not go into the details of this ordering for lack of space.

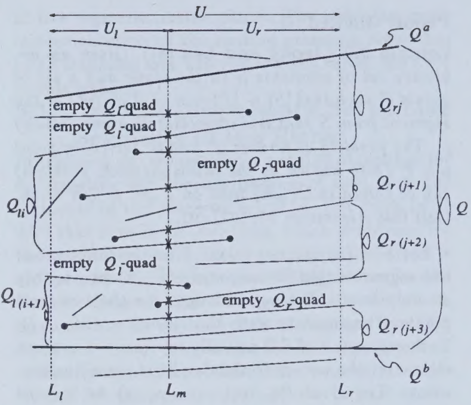


Figure 1: The Q_l-quads and the Q_r-quads of an active quad Q of U.

Once this is done, we can determine the empty Q_l-quads (Q_r-quads) by performing a segmented prefix scan operation that will count the number of endpoints from E_{U_l} (E_{U_r}) between every two consecutive spanning segments of U_l (U_r). Now the Voronoi diagrams of these empty quads can be computed and stored in the PEs easily. To construct the Q_l-diagram (Q_r-diagram) we need to merge together the Voronoi diagrams of *all* the Q_l-quads (Q_r-quads), empty as well as active, which can be done by "concatenating" the diagrams of all the Q_l-quads (Q_r-quads) in the correct sorted order (as in [22]). The above computation ensures that these diagrams are, in fact, already in the right order. Hence, the *horizontal merge* takes $O(\sqrt{k})$ time on $M_l \cup M_r$.

(3) **The horizontal merge:** In this final stage of the merge step, we obtain the Voronoi diagram of each active quad Q. This is done by merging the Q_l- and the Q_r-diagram, which involves the construction of the contour. The horizontal merge is the most complicated part of this algorithm. Once the contour is constructed, the Q_l-diagram to the left of the contour, the contour itself, and the Q_r-diagram to the right of the contour give us the final Voronoi diagram $Vor(S_U \cap Q)$ for every active quad Q of U. Our discussion will describe the computation performed for one active quad Q, with the assumption that the same steps are carried out for

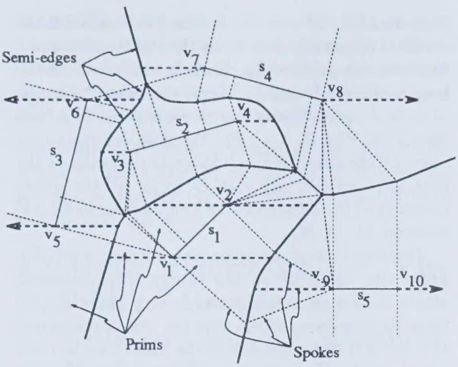


Figure 2: A Voronoi diagram augmented with spokes.

all the active quads of U.

As in the sequential methods of [9, 22] and the PRAM method of [7], we manipulate objects known as *primitive regions* for the construction of the contour. For the rest of this discussion, we will assume that the Q_l-diagram is augmented in the following way (the Q_r-diagram will be augmented in a similar way): For every element e (either a point or an open line segment) in $S_{U_l} \cap Q_l$, we add *spokes* [9] to the Voronoi region $Vor(e)$ of e. If v is a Voronoi vertex of $Vor(e)$, and if $v' = proj(v, e)$ (the projection of v on e), then the line segment obtained by joining v and v' is a spoke of $Vor(e)$. See Figure 2 for a Voronoi diagram augmented with spokes. In [7], the authors add some additional spokes. These additional spokes are indicated by bold dotted lines in Figure 2 (refer to [7] for details). All spokes define new sub-regions within $Vor(e)$. These sub-regions bounded by two spokes on two sides, part of e on one side, and a piece of Voronoi edge on the other side are called *primitive regions* (*prims* for short). The piece of Voronoi edge that forms one of the boundary edges of each prim is called a *semi-edge* [7]. Notice that since $VorSet(S_{U_l})$ consists of at most $O(k)$ Voronoi edges and vertices, the number of prims will also be $O(k)$. For the rest of this discussion, we will call the spokes of the Q_l-diagram as Q_l-spokes, the prims of the Q_l-diagram as Q_l-prims, and the semi-edges of the Q_l-diagram as Q_l-semi-edges (similarly for Q_r). The segment endpoints or open line segments that belong to $S_{U_l} \cap Q_l$ ($S_{U_r} \cap Q_r$) will be called Q_l-objects (Q_r-objects).

In the merge computation on the mesh so far, our technique has been to store a constant number of Voronoi edges per PE. Notice that each Voronoi edge (part of $B(e_1, e_2)$, say) actually defines two prims: one in each of the two Voronoi regions $Vor(e_1)$ and $Vor(e_2)$. So we will assume that both these prims are stored along with the Voronoi edge. It is easy to show that the additional spokes (mentioned above) that need to be added can be determined in constant time. The construction of the contour between the Q_l-diagram and the Q_r-diagram depends crucially on certain properties of the contour. These are stated below as lemmas (we refer the reader to [7, 22] for the proofs).

Lemma 3.3 (Goodrich et al. [7]) *Let α and β be Q_l- and Q_r-prims, respectively. Let $s_\alpha \in S_{U_l}$ and $s_\beta \in S_{U_r}$ be such that $\alpha \subseteq Vor(s_\alpha)$ and $\beta \subseteq Vor(s_\beta)$. Let $b_{\alpha,\beta} = B(s_\alpha, s_\beta) \cap \alpha \cap \beta$. If $b_{\alpha,\beta}$ is non-empty, then $b_{\alpha,\beta}$ defines a piece of the contour.*

Lemma 3.4 (Goodrich et al. [7]) *The contour is monotone with respect to the y-axis.*

Lemma 3.5 (Goodrich et al. [7]) *The contour intersects each spoke and each Voronoi semi-edge at most once.*

From the above lemmas it is easy to see that the contour intersects each prim in at most one continuous piece [7]. Before we proceed, we state an important lemma.

Lemma 3.6 *Given a set P of points in the plane, and the Voronoi diagram of a set S of line segments, where $|S| + |P| \leq n$. The problem of finding the Voronoi region that each point $p \in P$ lies in, can be solved in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh. Call this Algorithm* VorRegionLoc.

Algorithm VorRegionLoc can be implemented by using a technique similar to that given by Jeong and Lee [8] for Algorithm MultiLoc, with some minor modifications.

We now outline the important steps in the construction of the contour on the mesh. Notice that at this stage of the merge all the active quads of U are in sorted order in $M_l \cup M_r$, and within each such Q, we have the Q_l-diagram, followed by the Q_r-diagram. The contour consists of edges that are of the form $B(e_l, e_r)$, where e_l is a Q_l-object and e_r is a Q_r-object. Hence our goal is to identify all such pairs (e_l, e_r). Obviously, if a Q_l-object e_l is part of such pairs, then some of its Q_l-prims will be intersected by the contour (similarly for Q_r-objects). Notice (from Lemma 3.5) that determining if a prim is intersected by the contour is equivalent to determining if at least one of the spokes of that prim is intersected by the contour. Thus, we construct the contour as follows:

(a) Identify the Q_l-spokes that are intersected by the contour and arrange them in the order of intersection. Such an order exists because of the monotonocity property (Lemma 3.4) of the contour. Call this sorted list IS_l.

(b) Identify the Q_r-spokes that are intersected by the contour and arrange them in the order of intersection. Call this sorted list IS_r.

(c) From the two sorted lists IS_l and IS_r, determine the pairs $(e_l,\ e_r)$ such that $B(e_l,\ e_r)$ forms part of the contour (Lemma 3.3 helps us here).

A summary of the implementation of steps (a), (b) and (c) is given below:

Step (a): (i) *Identifying the Q_l-spokes that are intersected by the contour:* Every Q_l-spoke has one endpoint that is part of a Q_l-object. Obviously, this endpoint will always be closer to the Q_l-diagram than to the Q_r-diagram. However, the other endpoint (call this the "free" spoke-endpoint) of the Q_l-spoke may or may not be closer to the Q_l-diagram. If it is not, then the spoke will be intersected by the contour. Apply Algorithm VorRegionLoc, using the Q_r-diagram as the Voronoi diagram of the input, and the "free" endpoints of the Q_l-spokes as the point set P of the input. Clearly this can be done in $O(\sqrt{k})$ time on $M_l \cup M_r$. Consider a Q_l-spoke l'; l' is part of $Vor(e_l)$, say. Suppose the "free" endpoint p of l' lies in $Vor(e_r)$, where e_r is a Q_r-object. If $d(p,\ e_r) < d(p,\ e_l)$, then l' will be intersected by the contour. Since each PE has a constant number of Voronoi edges, we can now identify the intersected Q_l-spokes in constant time.

(ii) *Sorting the intersected Q_l-spokes:* We now arrange the intersected Q_l-spokes in the order in which they are intersected by the contour (from bottom to top, say). We find this order by *explicitly sorting* the spokes[2]. We will not go into the details of the ordering here for lack of space. Let this sorted list of spokes be called IS_l. IS_l can be found in $O(\sqrt{k})$ time on $M_l \cup M_r$.

Step (b): Analogous to steps (a)(i) and (a)(ii) above. Let the sorted list of intersected Q_r-spokes be called IS_r.

Step (c): Note that the sorted order of intersected Q_l-prims (Q_r-prims) is implicit in IS_l (IS_r): call this ordered set IP_l (IP_r). Consider some prim α form IP_l. We say that α *interacts* with prim $\beta \in IP_r$ if $b_{\alpha,\beta}$ (refer to Lemma 3.3) is nonempty. In general, α will interact with some subset of prims from IP_r. This subset will be a continuous interval of prims from IP_r [7]. Call this interval of prims I_α. Furthermore, all the prims that lie above α in IP_l can interact only with those prims of IP_r that lie above I_α [7].

We implement this step on the mesh in the following way. For every prim $\alpha \in IP_l$, we identify the topmost and bottommost prim of the interval I_α. Sequentially, this can be done by using binary search for each α. On the mesh, this step can be done by two applications of Algorithm SimultSrch (refer Lemma 3.1), which takes $O(\sqrt{k})$ time on $M_l \cup M_r$. Let P_t be the PE that holds the topmost prim of I_α and P_b be the PE that holds the bottommost prim of I_α. Each α can now find the length of the interval I_α. Next, we make $|I_\alpha|$ copies of α, and each of those copies reads $\beta \in IP_r$ from one of the PEs $P_t, \ldots, P_b$. We thus determine the piece of the contour $b_{\alpha,\beta}$. Since the lengths of the lists IP_l and IP_r are each $O(k)$ for all the active quads Q of U, the above step can be done in $O(\sqrt{k})$ time on $M_l \cup M_r$.

The run-time of the preprocessing step is $O(\sqrt{n})$. From the summary of the merge step described above, it is seen that the merge step takes $O(\sqrt{n})$ time. It therefore follows that the Voronoi diagram of a set of n line segments in the plane can be computed in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh. We state this result as a theorem.

Theorem 3.7 *The Voronoi diagram of a set of n nonintersecting (except possibly at endpoints) line segments in the plane can be found on a $\sqrt{n} \times \sqrt{n}$ mesh in $O(\sqrt{n})$ time (with no queueing).*

3.2 Motion Planning Using Voronoi Diagrams

In [15], Ó'Dúnlaing and Yap give a retraction method for planning the motion of an object (a disc) with two *dofs*, moving amongst polygonal obstacles[3]. They use the Voronoi diagram of the line segments that make up the obstacles to plan the motion of the object. We give the mesh-optimal parallel implementation of this method of motion planning. Let us assume that the object A has to be moved from point a to b. First we construct the Voronoi diagram and this takes $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh, as we have just shown. The next step is to remove all the Voronoi edges that do not satisfy the minimum clearance requirement. In other words, we want to delete all Voronoi semi-edges $v' = B(e_1, e_2)$ such that the minimum distance of the points on v' from e_1 and e_2 is less than some prespecified length r (the radius in the case of a moving disc). Clearly, assuming that we know r, this deletion can be done in constant time on the mesh, since each PE has a constant number of Voronoi edges. The remaining Voronoi edges define a graph which may be disconnected.

The next step is to find the Voronoi cells Vor_a and Vor_b that contain the points a and b, respectively. By Lemma 3.6, this can be done in $O(\sqrt{n})$ time. The last step is to find a path from an (undeleted) edge of Vor_a to an (undeleted) edge of Vor_b. One way to do this is by constructing the spanning tree and then finding this path, if one exists. In [3], Atallah and Hambrusch show that in a graph with edge set E, we can solve this problem in $O(\sqrt{|E|})$ time on a mesh with $|E|$ PEs. In the graph defined by the Voronoi diagram, $|E|$ is $O(n)$. Hence, it follows that we can implement the motion planning technique of [15] in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh, as stated below.

---

[2] In [7], the authors find the ordering of the spokes by creating a linked list and then using a list ranking algorithm on the PRAM. Such pointer manipulation is difficult to implement on the mesh, and so we avoid it.

[3] This particular retraction approach can actually be extended to the motion planning problem for any convex object with 2 *dofs* moving among convex polygonal obstacles [19].

Theorem 3.8 *Given a polygonal set of obstacles of size n, and a disc B, the motion of B from one position to another can be planned in $O(\sqrt{n})$ time on a $\sqrt{n} \times \sqrt{n}$ mesh.*

4 Mesh Algorithms for Visibility Graphs and the Related Motion Planning Problem

4.1 Visibility Graphs

We will now describe a mesh algorithm to compute the visibility graph of a given set of line segments in the plane. As noted in the earlier sections, the efficient construction of the visibility graph is an important substep in motion planning. To our knowledge, this problem has not been solved on the mesh. We will show that, given an input set S ($|S| = N$) of nonintersecting line segments in the plane, we can identify mesh-optimally all the segment vertices that are visible from a given point p in $\theta(\sqrt{n})$ (where $n = 2N$) time on a $\sqrt{n} \times \sqrt{n}$ mesh. This will immediately give us an algorithm for constructing the visibility graph, G_S.

Let $S = \{s_0, s_1, \ldots, s_{N-1}\}$ be the input set of line segments that do not intersect (except possibly at endpoints), and let p be the point from which we want to determine visibility. Let v_{2i} and v_{2i+1} (we will assume $x(v_{2i}) < x(v_{2i+1})$) be the two endpoints of segment s_i. The visibility from a point problem is to determine that part of the plane that is visible from p, assuming that every segment is opaque. Notice that this is equivalent to identifying those vertices v_i that are "seen" from p. As in [4], we will assume, without loss of generality, that p is a point at $-\infty$. This is only to make the description of the algorithm simpler. The case when p is not at infinity is a straightforward adaptation of this algorithm. Since p is at $-\infty$, to compute the visibility from p, we need to compute the *lower envelope* of the set of segments in S [4]. The lower envelope is the collection of those segment parts that can be seen from below.

In [4], the authors give a PRAM algorithm that uses the cascading divide-and-conquer technique for solving the visibility from a point problem. Along the same lines, we will describe a recursive algorithm for computing the lower envelope on the mesh. We will first describe the merge step and then give the details of the mesh algorithm. Let S_1 be the set consisting of half the elements of S, and let S_2 contain the other half. Suppose that we have recursively computed the lower envelopes of S_1 and S_2. The lower envelope of the segments in S_i $(i = 1, 2)$ is available to us in the following manner: The endpoints of the segments in S_i have been sorted according to their x-coordinates (for the sake of simplicity, let us assume that no two endpoints have the same x-coordinate). In this sorted list (call it V_i), assume that a vertical line is placed through each endpoint. This divides the plane into vertical strips of region called *slabs* (call these the V_i-slabs). The recursive computation gives, for every V_i-slab, the segment of S_i that is visible from below (i.e. is part of the lower envelope) in that slab. Now, we want to merge these two envelopes to form the final lower envelope. First merge V_1 and V_2 to form V. The set V defines a new set of slabs. Each V-slab (say u) lies within some unique V_1-slab (say u_1) and some unique V_2-slab (say u_2). Note that u could, in fact, be the same as either of u_1 or u_2. Let s_1 and s_2 be the recursively computed lower envelope segments in the slabs u_1 and u_2, respectively. Then, the segment of S that is visible from below in u is nothing but the lower of s_1 and s_2 (note that such an ordering is uniquely defined on the two segments).

The algorithm for computing the lower envelope (i.e. visibility from $-\infty$) is given below.

Algorithm VisFromPoint;

Input: The endpoints are distributed one per processor on a $\sqrt{n} \times \sqrt{n}$ mesh with the shuffled row-major indexing scheme. The PE P_j ,$j \in \{0, 1, \ldots, n-1\}$ has endpoint v_j and also the segment that v_j is an endpoint of.

Output: The endpoints will be in sorted order on the mesh. Thus each PE P_i is associated with a slab in the obvious way. P_i will also have the segment s that is part of the lower envelope (i.e. is visible) in that slab.

Initialization: Every PE P_i has the following fields as part of its record: endpoint initialized to v_i; lowerseg, which contains, at any stage, the lowest segment (found up to that stage) for the slab defined by P_i ; whichblock, which indicates (for the merge step) whether an endpoint came from the left block or the right.

Base Step: lowerseg is set to the segment $s_{i/2}$ if i is even and to $\emptyset$ otherwise. Let S_1 be the subset of segments of S in the left block, and S_2 be the subset in the right block.

Recursive Step: Solve recursively in parallel using S_1 for S in the left block and S_2 for S in the right block.

Merge Step:

(i) Set whichblock to 0 if P_i belongs to the left block and to 1 if it belongs to the right block.

(ii) Merge the two sets S_1 and S_2 according to the endpoint field.

Note: We now need to update the lowerseg field in each PE. As explained earlier, every new slab u

of the merged set needs to compare the recursively computed **lowerseg** fields of the two old slabs u_1 and u_2 that it is a part of. This can be done by using the *selected broadcasting* operation as stated below.
(iii) The subset of elements that needs to be broadcast is the **lowerseg** field in every processor with **whichblock**= 0. Let $\{l_1, l_2, \ldots, l_{n/2}\}$ (where $n = 2|S|$) be the set of these **lowersegs** in sorted order and let I_{l_i} be the index of the processor in which l_i resides. The *selected broadcasting* operation will send l_i, $1 \leq i \leq n/2$ to every PE from $P_{I(l_i)}$ to $P_{I(l_{i+1})-1}$. Put l_i in a local register called **lowerseg1**.
(iv) Similar to step (iii), except that the broadcast elements are the **lowerseg** fields from processors with **whichblock**= 1. Here, the broadcast element is put in a local register called **lowerseg2**.
(v) Every PE updates the **lowerseg** field to the lower of **lowerseg1** and **lowerseg2**.
It is clear that the merge step takes $O(\sqrt{n})$ time and thus we have the following theorem.

Theorem 4.1 *Algorithm* VisFromPoint, *which computes the lower envelope of a set of segments S, runs in $O(\sqrt{n})$ time (with no queueing) on a $\sqrt{n} \times \sqrt{n}$ mesh, where $|S| = n/2$.*

Notice that the computation of the lower envelope on the mesh immediately tells us which endpoints of S are visible from $-\infty$. When the point p is not at $-\infty$, the algorithm is the same as above, except that instead of merging the endpoints of the line segments according to their x-coordinate, we merge them according to the polar angle that they make with p (measured with respect to some fixed axis). In order to construct the entire visibility graph, we can use the above algorithm in a straightforward way. When a vertex v_i is used as p, we can obtain the set of vertices of S that are visible from v_i. In other words, we know which nodes are adjacent to the node corresponding to v_i in the visibility graph. If we repeat this for every endpoint v_j, in parallel, we can construct the visibility graph of a set S of segments in $\theta(\sqrt{n})$ time using n^2 processors (i.e. n of the $\sqrt{n} \times \sqrt{n}$ meshes). This is optimal since the visibility graph may have $O(n^2)$ edges in the worst case, and we will need n^2 processors to represent the graph (under the assumption that each processor has only a constant amount of storage).

4.2 Motion Planning Using Visibility Graphs

Lozano-Pérez and Wesley [12] give an approximate projection method for planning the motion of a convex object B (of constant size) with two *dofs*, moving between convex obstacles (total size n). We summarize their sequential approach briefly: First "expand" each convex obstacle O according to some reference point on B, which can be done in time proportional to the size of O. B will not collide with O if and only if the reference point of B lies outside of the expansion of O. Let A be the union of all the expanded obstacles. Since B has 2 *dofs*, the configuration space of B is 2-dimensional. In fact, the complement of A in the plane is the set of free configurations, FP, for B. Let E be the set of edges of A. The next step is to compute the *visibility graph* of the set of edges E. The visibility graph is precisely the connectivity graph (of the projection method) that we are looking for. In addition, we have the useful property that the shortest possible path for B between two points in the plane while avoiding the obstacles is given by the shortest path between the corresponding two nodes in the visibility graph (where the edge weight is the Euclidean length of the edge). Thus we can find a shortest path for B by performing a shortest-path graph search on the visibility graph. The sequential run-time of this motion planning method is $O(n^2)$.

Assume the obstacle set is stored in a $\sqrt{n} \times \sqrt{n}$ mesh. First we expand the obstacles according to the moving object B: We relay the information about B to each of the PEs in $\sqrt{n}$ time. The expansion of all the obstacles can be done in at most $O(\sqrt{n})$ time. Note that these expanded obstacle edges might now intersect with each other. When the obstacles are convex, it can be shown that the number of such intersections can be at most $O(n)$ [19]; thus the new obstacle edge set will also be $O(n)$. We can also find the new obstacle edges by using a brute force technique which is very inefficient, but will not affect the run-time of this motion planning algorithm on the mesh. We can simply compute the intersection of every edge of the expanded obstacle set with the other edges of that set. This will give us the new edge segments, and can be computed in $O(n)$ time on a mesh with n PEs.

We now have to make n copies of the new obstacle edge set on n sets of $\sqrt{n} \times \sqrt{n}$ meshes so that we may compute visibility from each of the n endpoints. These copies can clearly be made in $O(n)$ time on a mesh with n^2 processors. We know, as mentioned above, that the visibility information from each endpoint can be computed in $O(\sqrt{n})$ time by using Algorithm VisFromPoint on each of these submeshes.

Suppose that the object B has to be moved from point a to point b. First we establish the visibility information from a and b, which can be done in $O(\sqrt{n})$ time using Algorithm VisFromPoint. We can compute the shortest path from a to b by solving the all-pairs shortest path problem for the visibility graph, using the euclidean length of the edges as the corresponding edge weights[4]. We first find the adjacency matrix representing the visibility graph, which can be done in $O(n)$ time (a sorting step) on the mesh of size n^2. Then we find the all-pairs shortest paths in $O(n)$ time by using a pipelining technique (see [11]). Thus, planning the motion of a convex object of two *dofs* moving among convex obstacles can be done in $O(n)$ time on a $n \times n$ mesh. Even though this mesh algorithm is not very work-efficient when compared to the $O(n^2)$ sequential algorithm, note that this is the best we can do since we will need n^2 PEs to represent the adjacency matrix.

---

[4] Note that, for our purposes here, solving the single-source shortest path (from a) problem would have sufficed. However, there are no known optimal parallel

5 Conclusions

We have given efficient parallel algorithms for some important geometric problems on the mesh-connected-computer. As a consequence, we obtained efficient parallel motion planning algorithms for some fundamental special cases. The development of parallel algorithms for the interesting and complex geometric problems that are of relevance can lead to significantly faster solutions for motion planning problems. Moreover, different parallel techniques for such problems could lead to new insights into planning motion. For example, there is no known optimal PRAM algorithm for the construction of the Voronoi diagram of line segments. Numerous problems in computational geometry that have no known optimal deterministic parallel algorithms have yielded to techniques such as randomization. In particular, it is possible that randomization could lead to a better parallel algorithm for the construction of the Voronoi diagram of line segments. In addition, it would also be interesting to see if randomization can be a useful strategy for designing faster solutions to various special cases of motion planning.

Acknowledgments: We would like to thank Norm Badler for his generous support and encouragement.

References

[1] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. K. Yap. Parallel Computational Geometry. *Algorithmica*, 3:293–327, 1988.

[2] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai. Visibility of Disjoint Polygons. *Algorithmica*, 1986.

[3] M. Atallah and S. Hambrusch. Solving Tree Problems on a Mesh-connected Processor Array. *Information and Computation*, 69(1-3):168–187, 1986.

[4] M. J. Atallah, R. Cole, and M. T. Goodrich. Cascading Divide-and-Conquer: A Technique for Designing Parallel Algorithms. *SIAM J. Comput.*, 18(3):499–532, June 1989.

[5] J. Canny. *Complexity of Robot Motion Planning*. PhD thesis, MIT, 1987.

[6] F. Dehne. Solving Visibility and Separability Problems on a Mesh-of-processors. *The Visual Computer*, 3(6):356–370, 1988.

[7] M. T. Goodrich, C. Ó'Dúnlaing, and C. K. Yap. Constructing the Voronoi Diagram of a Set of Line Segments in Parallel. In *Lecture Notes in Computer Science: 382, Algorithms and Data Structures, WADS*, pages 12–23. Springer-Verlag, 1989.

[8] C. S. Jeong and D. T. Lee. Parallel Geometric Algorithms on a Mesh-Connected Computer. *Algorithmica*, 5(2):155–177, 1990.

[9] D. G. Kirkpatrick. Efficient Computation of Continuous Skeletons. In *Proc. 20th IEEE Symp. on FOCS*, pages 18–27, 1979.

[10] D. T. Lee and R. L. Drysdale. Generalization of Voronoi Diagrams in the Plane. *SIAM J. Comput.*, 10(1):73–87, February 1981.

[11] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1992.

[12] T. Lozano-Pérez and M. A. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *CACM*, 22(10):560–570, 1979.

[13] R. Miller and Q. F. Stout. Mesh Computer Algorithms for Computational Geometry. *IEEE Transactions on Computers*, 38(3):321–340, 1989.

[14] C. Ó'Dúnlaing, M. Sharir, and C. K. Yap. Retraction: A New Approach to Motion-Planning. In J. E. Hopcroft, J. T. Schwartz, and M. Sharir, editors, *Planning, Geometry and Complexity of Robot Motion*, chapter 7, pages 193–213. Ablex Pub. Co., Norwood, N.J., 1987.

[15] C. Ó'Dúnlaing and C. K. Yap. A 'Retraction' Method for Planning the Motion of a Disc. *J. Algorithms*, 6:104–111, 1985.

[16] J. H. Reif and S. Sen. Polling: A New Randomized Sampling Technique for Computational Geometry. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 394–404, 1989.

[17] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Adv. in Appl. Math*, 4:298–351, 1983.

[18] M. I. Shamos. Geometric Complexity. In *Proc. 7th ACM STOC*, pages 224–233, 1975.

[19] M. Sharir. Efficient Algorithms for Planning Purely Translational Collision-Free Motion in Two and Three Dimensions. In *Proc. IEEE Symp. on Robotics and Automation*, pages 1326–1331, Los Alamitos, Calif., 1987. CS Press.

[20] M. Sharir. Algorithmic Motion Planning in Robotics. *IEEE Computer*, March 1989.

[21] E. Welzl. Constructing the Visibility Graph for n Line Segments in $O(n^2)$ Time. *Info. Proc. Letters*, 20:167–171, 1985.

[22] C. K. Yap. An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. *Discrete and Computational Geometry*, 2:365–393, 1987.

---

algorithms for this problem.

Improved Recursive Bisection Line Drawing Algorithms

Phil Graham, S. Sitharama Iyengar, and Si-Qing Zheng

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract. Fractals have recently been used in computer graphics to draw self-similar objects in nature such as coastlines and mountains. They are also used to draw self-similar images such as fractal curves and trees. Since lines are self-similar and it has been shown that fractals are useful in drawing other self-similar objects, the development of fractal line drawing algorithms apprears to be a worthwhile endeavor, not only for practical reasons, but also to gain greater insight into techniques which can improve the performance of other fractal algorithms. In this paper, we present modifications which reduce the logarithmic space requirements of an existing fractal line drawing algorithm to a constant while increasing its speed. The resulting algorithm is faster than traditional line drawing algorithms, especially when the lines being drawn are at or near horizontal, diagonal, and vertical.

1 Introduction

Over the years, much attention has been devoted to developing efficient line drawing algorithms in computer graphics. In addition to drawing lines and modeling geometric shapes, such as squares, line drawing algorithms are used to approximate other shapes such as circles. They are also used in ray tracing and other applications involving lines.

Truly, line drawing algorithms are a fundamental topic in the area of graphics algorithms. Various algorithms have been presented which eliminate or greatly reduce the need to perform multiplications and divisions. For instance, the traditional Bresenham algorithm [1] performs integer logic on the error term with respect to one coordinate as values of the other coordinate are incremented. The algorithm in [11] uses integer logic to generate line points by instead using a double step increment with respect to one of the coordinate axes. Bresenham's run length slice algorithm [2] calculates a slice of movements with respect to a particular coordinate axis in each iteration of a loop. Other approaches to speeding up the scan-conversion process are based on Euclid's algorithm [4, 5]. There are also line drawing algorithms which are derivations or variations of these algorithms [3].

All of the algorithms just described always set the pixel closest to the line being drawn (we will call this the *true line*, as opposed to the line which is formed by setting pixels). However, this degree of precision is not necessary in all applications, especially on high resolution devices. In real-time or interactive graphics, faster algorithms which draw approximations to lines are often preferable, as long as they keep the error of the pixels being set within a reasonable limit. Recently, Rankin proposed a method called the *recursive bisection* (RB) algorithm [10] which uses a fractal approach [9] to draw line approximations. In [10], it is shown that the algorithm gives good approximations on low resolution devices. On high resolution devices, the difference between a line drawn by the RB algorithm and one drawn by setting the pixel closest to the true line is imperceptible. In many cases, the recursive bisection method is faster than the traditional Bresenham algorithm. Moreover, [6] shows that there are advantages in using a parallel version of the algorithm in an MIMD environment. Therefore, the development of algorithms which draw line approximations is both an interesting and practical matter.

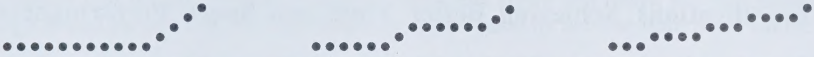
In this paper, modifications to the recursive bisection method of drawing lines are presented which significantly improve its performance with respect to both time and space. We also derive a formula for finding the maximum distance of any pixel from the true line for various classifications of lines and discuss a way to reduce this distance while adding virtually no additional computation time.

2 Overview of the RB Algorithm

In this paper, as in [10], standard notation relating to line algorithms is used whenever possible. (x_0, y_0) refers to the x and y coordinates of the starting point, and (x_1, y_1) refers to the x and y coordinates of the endpoint. Various horizontal and vertical components of the line are defined as follows:

$$u = x_1 - x_0,$$
$$v = y_1 - y_0, \text{ and}$$
$$w = u - v.$$

The slope is a floating point number and is determined by the equation $m = v / u$. Also, for the remainder of this paper, it is assumed that $w \geq v > 0$ in all lines being drawn in order to simplify discussions. Any of the other cases are handled in a similar fashion.



(i) Initially we have $S^{10}D^4$.

(ii) After one application of the replacement rule, we have $S^5D^2S^5D^2$.

(iii) After applying the replacement rule to each SD pair in (ii) we have $S^2D^1S^3D^1S^2D^1S^3D^1$.

Figure 1. The lines that result from various applications of the replacement rule when $w = 10$ and $v = 4$.

The recursive bisection line drawing algorithm uses the symbols S and D to represent sideways and diagonal movements when drawing the line from point (x_0, y_0) to point (x_1, y_1). In short, it defines lines fractally with the recursive replacement rule:

$$S^w D^v \rightarrow S^{wl} D^{vl} S^{wr} D^{vr}$$

where $wl = w \text{ div } 2$, $vl = v \text{ div } 2$, $wr = w - wl$, and $vr = v - vl$. Of course, "div" is defined to be integer division, as it is in the Pascal language. The replacement rule is no longer applied when either w or v equals 0 or 1. Figure 1 gives an example of the lines that result from applications of the replacement rule when $w = 10$ and $v = 4$.

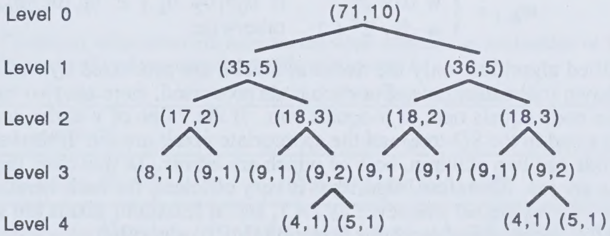


Figure 2. An example of an SD-tree for $w = 71$ and $v = 10$. In the diagram, the numbers in parentheses show the values of w and v at each node. The resulting run-length sequence is $S^8D^1S^9D^1S^9D^1S^4D^1S^5D^1S^9D^1S^9D^1S^9D^1S^4D^1S^5D^1$.

As shown in [10], the solution generated by the RB method for any line parameter can be characterized as a binary tree (Figure 2). We will call this tree the SD-partition tree (or simply the SD-tree). Any subtrees of the SD-tree will be called an SD-subtree. Throughout this paper, h is used as a shorthand notation for $\lfloor \log_2 v \rfloor$, and k is any integer such that $0 \leq k \leq h$. Also, the nodes at a given level k of the SD-tree (with the root at level 0) are numbered 0, 1, 2, ..., $2^k - 1$ from left to right. The definitions below are used in the algorithms and proofs which follow.

Definition 1: $n_{k,\,i}$ denotes node number i in level k of the SD-tree.
Definition 2: $n_{k+1,\,2i}$ ($n_{k+1,\,2i+1}$) denotes the left (right) child of $n_{k,\,i}$ (these nodes may not exist if $k = h$).
Definition 3: The values of v (w) at $n_{k,\,i}$ are denoted as $v_{k,\,i}$ ($w_{k,\,i}$). The values of v and w at the nodes in level $h + 1$ are denoted in a similar fashion (we add this note because $0 \le k \le h$).
Definition 4: $D\_ops_{w,\,k,\,i}$ ($S\_ops_{v,\,k,\,i}$) is the number of D (S) operators that occur to the left of the SD-subtree rooted at $n_{k,\,i}$.
Definition 5: Any given line is called ***h*-complete** since the SD-tree corresponding to the line is full down to level h and no further.

In addition, the n-bit binary representation of a given variable, say v, is denoted as $b^v = b^v_{n-1}b^v_{n-2}b^v_{n-3}\ldots b^v_0$, where b^v_{n-1} is the most significant bit. The binary representations of other variables are denoted in a similar fashion.

3 Modifications Achieving Better Time and Space Performances

Other than the root node, all nodes of the SD-tree correspond to recursive calls to the procedure. That is, for a given pair of v and w values, the original recursive bisection algorithm "traverses" the SD-tree and then sets pixels whenever a leaf is encountered. Since we are only interested in the values of v and w at the leaves of the SD-tree and space is needed to store the activation record for each recursive call, it is obvious that an algorithm that can quickly find the values of v and w at the leaf nodes without determining the values at the other levels will be more efficient, with respect to both time and space. This is the basic idea of our algorithm. It is achieved by making use of the following theorems which are proven in [6]. In these theorems, $\overline{b^i_0b^i_1b^i_2\ldots b^i_{k-1}}$ is the complement of $b^i_0b^i_1b^i_2\ldots b^i_{k-1}$.

Theorem 1: If $1 \le k \le h$, then

$$v_{k,\,i} = \begin{cases} v \text{ div } 2^k & \text{if } \overline{b^i_0b^i_1b^i_2\ldots b^i_{k-1}} \ge b^v_{k-1}b^v_{k-2}b^v_{k-3}\ldots b^v_0; \\ v \text{ div } 2^k + 1 & \text{otherwise.} \end{cases}$$

Theorem 2: If $1 \le k \le h$, then

$$w_{k,\,i} = \begin{cases} w \text{ div } 2^k & \text{if } \overline{b^i_0b^i_1b^i_2\ldots b^i_{k-1}} \ge b^w_{k-1}b^w_{k-2}b^w_{k-3}\ldots b^w_0; \\ w \text{ div } 2^k + 1 & \text{otherwise.} \end{cases}$$

In our modified algorithm, only the nodes at level h are processed by making the simple comparisons shown in the theorems. For each node processed, there are two cases: either the value of v at the node equals one, or it equals two. If the value of v at the node equals one, then the node is a leaf in the SD-tree, and the appropriate pixels are set. If the value of v equals two, then the node has two children, both of which are leaves. In this case, the pixels for the two child nodes are set. Therefore, algorithm is very efficient; for each iteration of the main loop, at least two pixels are set whenever $v_{h,\,i} = 1$, and at least four pixels are set when $v_{h,\,i} = 2$. We will call this the *modified recursive bisection* (MRB) algorithm .

Naturally, when implemented, the speed of any algorithm is highly instruction dependent. Therefore, in the discussion which follows, it is assumed that optimal time-equivalent instructions were used for finding h and $\overline{b^i_0b^i_1\ldots b^i_{h-1}}$ in order to show the potential speed of the modified RB algorithm. With this said, the run-time performance of the MRB algorithm was compared with a textbook version of Bresenham's algorithm [8] and the recursive bisection algorithm using compiled C on a DECstation. The lines which were tested had a value of v which ranged from 30 to 60 units in length, and the w values ranged from v to $200 + v$. The results show that our algorithm is from 2.2 to 4.5 times as fast as the RB algorithm and from 20 to 45% faster than Bresenham's algorithm. In both cases, the relative speed of the MRB algorithm depends on the values of $w_{h,\,i}$. When compared to the RB algorithm, the speed of the algorithm depends on the depth of the SD-tree as well. Although the MRB algorithm was faster for all the lines we chose, we do not claim that our algorithm is faster for every possible line. Since there are several additional statements during startup, the MRB will be slightly slower when a line is only a few pixels long.

4 Error Analysis

As noted in the introduction, the difference between a line drawn by the RB algorithm and one drawn by setting the pixel closest to the true line is imperceptible on higher resolution devices. [10] shows that the lines produced by the RB algorithm are highly regular. Although examples were also given which show that the sum of the errors for each pixel set in the RB algorithm can be comparable to that for the traditional Bresenham algorithm, no bounds on the error were given. In the following subsections, a formula for the maximum possible distance of any pixel from the true line for the RB algorithm is obtained. Of course, these discussions also hold for the MRB algorithm since it sets the same pixels as the RB algorithm.

Since the pixels set can lie above, on, or below the true line, one must consider the possibilities that the pixel having maximum error lies above and below the true line when setting a bound on the error. We will derive an upper bound by the four steps listed below:

1) formulate the functions defining $S\_ops_{w,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$,
2) use the functions for $S\_ops_{w,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$ to derive a function for the maximum possible error of a pixel which is dependent on the values of h, v, and i,
3) determine the values of v and i which maximize the error function for all values of h, and
4) express the value of the maximum error of a pixel in a form which is easy to compute.

Before proceeding further, we shall give more precise definitions of error and finding the maximum possible error of any pixel. In this paper the *error* will be defined as the difference between the y coordinate of the point on the line and the y coordinate of the pixel set at a given coordinate of x. It is denoted as ε. Then, by definition, the error of a pixel set above the true line has a negative value and the error of those pixels set below the true line has positive value. Of course, the error equals zero for pixels lying on the true line. The problem of *finding the maximum possible error of any pixel* is then to find the maximum possible value of $|\varepsilon|$ for any pixel set by the RB algorithm.

Table 1. Conditions when terms are incremented when determining the number of S operators processed before the SD-subtree rooted at the *i*th node in level *k*.

Term	Condition When Term is Incremented
$t_0 = w \text{ div } 2^1$	never
$t_1 = w \text{ div } 2^2$	$(b^w_1 b^i_{k-2} = 1)$ and $(\overline{b^i_{k-1}} < b^w_0)$
$t_2 = w \text{ div } 2^3$	$(b^w_2 b^i_{k-3} = 1)$ and $(\overline{b^i_{k-2} b^i_{k-1}} < b^w_1 b^w_0)$
$t_3 = w \text{ div } 2^4$	$(b^w_3 b^i_{k-4} = 1)$ and $(\overline{b^i_{k-3} b^i_{k-2} b^i_{k-1}} < b^w_2 b^w_1 b^w_0)$
...	...
$t_{k-1} = w \text{ div } 2^k$	$(b^w_{k-1} b^i_0 = 1)$ and $(\overline{b^i_1 b^i_2 \ldots b^i_{k-1}} < b^w_{k-2} b^w_{k-3} \ldots b^w_0)$

4.1 Define Functions for $S\_ops_{w,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$

When determining the maximum possible error, it is desirable to be able to write $D\_ops_{v,\,h,\,i}$ and $S\_ops_{w,\,h,\,i}$ in a closed form because the pixel corresponding to the last operator at $n_{h,\,i}$ has no error if:

$$\frac{D\_ops_{v,\,h,\,i+1}}{D\_ops_{v,\,h,\,i+1} + S\_ops_{w,\,h,\,i+1}} = m.$$

Naturally, the relationship shown above also indicates whether the pixel corresponding to the last operator is above or below the true line. Therefore, it is necessary to determine how to write $S\_ops_{w,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$ in some type of mathematical expression. $S\_ops_{w,\,k,\,i}$ and $D\_ops_{v,\,k,\,i}$ in the discussions below refer to general solutions of the problem for any node i at level k, where $0 \leq k \leq h$.

As shown in [6], $S\_ops_{w,\,k,\,i}$ is found as follows:

$$S\_ops_{w,\,k,\,i} = \sum_{y=0}^{k-1} (t_y + inc_y)$$

where

$$t_y = \begin{cases} w \text{ div } 2^{y+1} & \text{if } b^{i}_{k-1-y} = 1; \\ 0 & \text{if } b^{i}_{k-1-y} = 0 \end{cases}$$

and inc_y is either 0 or 1, depending on whether the conditions shown in Table 1 are satisfied. Although the problem is reducible to multiplication, there seems to be no easy way to write the above equality in a simple closed mathematical form. Therefore, an upper bound for $S\_ops_{w,\,k,\,i}$ will be determined, and the value of $S\_ops_{w,\,k,\,i}$ will be set equal to this upper bound minus a correction factor. Upon inspecting the replacement rule, one may suspect that an upper bound for $S\_ops_{w,\,k,\,i}$ is (iw div 2^k) and that an upper bound for $D\_ops_{v,\,k,\,i}$ is (iv div 2^k) since it is computed in a similar manner. The theorems below state that this is indeed the case. Formal proofs are omitted, instead given in [7], due to limitations in space.

Theorem 3: The value of $S\_ops_{w,\,k,\,i}$ is less than or equal to (iw) div 2^k.

Theorem 4: The value of $D\_ops_{v,\,k,\,i}$ is less than or equal to (iv) div 2^k.

By Theorem 3, the number of S operators processed before $n_{h,\,i}$ is less than or equal to (iw div 2^h). It follows that:

$$S\_ops_{w,\,h,\,i} \leq \frac{iw}{2^h}.$$

Subtracting a correction factor, c, which is a function of w, h, and i, we have:

$$S\_ops_{w,\,h,\,i} = \frac{iw - c_{w,\,h,\,i}}{2^h}$$

where $c_{w,\,h,\,i} \geq 0$. Using a similar argument, $D\_ops_{v,\,h,\,i}$ is defined as follows:

$$D\_ops_{v,\,h,\,i} = \frac{iv - c_{v,\,h,\,i}}{2^h}$$

where $c_{v,\,h,\,i} \geq 0$. Since v and w are distributed in the SD-tree in the same fashion, both $c_{w,\,h,\,i}$ and $c_{v,\,h,\,i}$ have the same range of values.

4.2 Derive Function for Maximum Error of a Pixel

The above definitions of $S\_ops_{w,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$ will now be used to define the maximum possible error of a pixel in terms of $c_{w,\,h,\,i}$ and $c_{v,\,h,\,i}$. First, we will make the observation that it is not necessary to examine all $u + 1$ pixels set in any given line to determine the maximum error. Only the leftmost and rightmost pixels at each level of y from y_0 to y_1 need to be examined. This number can be reduced further by noting that the leftmost pixel set at y_0 is the starting point and that there is only one pixel set at y_1, the endpoint. Of course, both the starting and ending points lie on the true line, have zero error, and can be ignored. Therefore, only $2v - 1$ pixels must be examined in order to determine the maximum error of a given line. Of these, we are only concerned with the leftmost pixel set at each level of y when determining the maximum possible error above the line and the rightmost pixel set at each level of y when determining the maximum possible error below the line. These observations are used in the following lemmas and theorem which gives an upper bound on the error for all lines having a given range of values for v by finding the maximum distance of any pixel above and below any h-complete line, then showing that the distance below the line is always the greater of the two values.

Lemma 1: The maximum distance of any pixel set above an h-complete line is less than $\left|\min\left(\frac{-c_{w,\,h,\,i}}{2^{h+1}} - 1\right)\right|$.

Proof: In this proof, the only pixels of interest are the ones which correspond to D operators because the pixel having maximum error above the true line is the leftmost one set at a given level of y. Since the values of v at $n_{h,\,i}$ may equal one or two, two different cases will be considered when determining the maximum possible error at the node. If the value of v at node number i in level h equals one, then the error of the pixel associated with the D operator at $n_{h,\,i}$ is as follows:

$$
\begin{aligned}
\varepsilon &= (\text{error of pixel associated with last D operator at } n_{h,\,i}) \\
&= (y \text{ coordinate of point on line}) - (y \text{ coordinate of pixel set}) \\
&= m(D\_ops_{v,\,h,\,i+1} + S\_ops_{w,\,h,\,i+1}) - D\_ops_{v,\,h,\,i+1} \\
&= \frac{v}{v+w}(D\_ops_{v,\,h,\,i+1} + S\_ops_{w,\,h,\,i+1}) - D\_ops_{v,\,h,\,i+1} \\
&= \frac{vS\_ops_{w,\,h,\,i+1} - wD\_ops_{v,\,h,\,i+1}}{v+w} \\
&= \frac{v\dfrac{iw - c_{w,\,h,\,i+1}}{2^h} - w\dfrac{iv - c_{v,\,h,\,i+1}}{2^h}}{v+w} \\
&= \frac{-vc_{w,\,h,\,i+1} + wc_{v,\,h,\,i+1}}{2^h(v+w)} \\
&\geq \frac{-vc_{w,\,h,\,i+1}}{2^h(v+w)}
\end{aligned}
$$

If the value of v at node number i in level h equals two, then the values of the maximum error of the pixels associated with the D operators of the child nodes of $n_{h,\,i}$ must be determined. The maximum error of the pixel associated with the D operator at $n_{h+1,\,2i+1}$ is identical to that for the case where v equals one. The error of the pixel associated with the D operator at $n_{h+1,\,2i}$ is:

$$
\begin{aligned}
\varepsilon &= (\text{error of last D operator at } n_{h,\,i-1}) + (\text{error for processing operators at } n_{h+1,\,2i}) \\
&= \frac{-vc_{w,\,h,\,i} + wc_{v,\,h,\,i}}{2^h(v+w)} - 1 + m(w_{h+1,\,2i} + 1) \\
&> \frac{-vc_{w,\,h,\,i}}{2^h(v+w)} - 1.
\end{aligned}
$$

Obviously, the minimization of the above function describes a pixel having greater distance above the true line than the earlier function for ε. Since $w \geq v$, the proof of the lemma is concluded by substituting v for the value of w in the function above. ☐

Lemma 2: The maximum distance of any pixel set below an h-complete line is less than $\max\left[\max\left(\frac{c_{v,\,h,\,i}}{2^h} + mw_{h+1,\,2i}\right), \max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right)\right]$.

Proof: The proof is similar to that for Lemma 1. It is omitted for brevity. ☐

Lemma 3: If a line is h-complete, then the value of $mw_{h+1,\,2i}$ is less than one.

Proof: The lemma is proven by contradiction. Assume that $mw_{h+1,\,2i} \geq 1$. Since the values of w at level $h + 1$ equal either (w div 2^{h+1}) or [(w div 2^{h+1}) + 1], the following statements must hold:

$$\begin{aligned}
&m[(w \text{ div } 2^{h+1}) + 1] \geq 1,\\
&m(w \text{ div } 2^{h+1}) \geq 1 - m,\\
&\frac{v(w \text{ div } 2^{h+1})}{v + w} \geq \frac{w}{v + w},\\
&v(w \text{ div } 2^{h+1}) \geq w,\\
&v \geq \frac{w}{(w \text{ div } 2^{h+1})},\\
&v \geq \frac{w}{(w / 2^{h+1})}, \text{ and}\\
&v \geq 2^{h+1}.
\end{aligned}$$

However, v cannot be greater than or equal to 2^{h+1} because the line is h-complete. Since assuming that $mw_{h+1,\,2i} \geq 1$ leads to a contradiction, the lemma must hold. ☐

Theorem 5: The maximum error in an h-complete line is less than $\max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right)$.

Proof: From Lemmas 1 and 2, the maximum error in an h-complete line is less than $\max\left[\max\left(\frac{c_{v,\,h,\,i}}{2^h} + mw_{h+1,\,2i}\right),\ \max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right),\ \left|\min\left(\frac{-c_{w,\,h,\,i}}{2^{h+1}} - 1\right)\right|\right]$. The theorem will be proven by showing that the second term is always greater than or equal to the first and third terms above. Clearly, $\left|\min\left(\frac{-c_{w,\,h,\,i}}{2^{h+1}} - 1\right)\right| \leq \max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right)$. As shown in Lemma 3, $mw_{h+1,\,2i} < 1$. Therefore, $\max\left(\frac{c_{v,\,h,\,i}}{2^h} + mw_{h+1,\,2i}\right) < \max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right)$, and the maximum error must always be less than $\max\left(\frac{c_{v,\,h,\,i}}{2^h} + 1\right)$. ☐

In Theorem 5 it is proven that bounds on the maximum error can be set when lines are classified according to the level at which their SD-tree is complete. Since w is assumed to approach infinity, these bounds will only be approached for longer lines. However, these bounds are tight in theory because the term which has been ignored in Lemma 2, $- vc_{w,\,h,\,i+1}$, is in fact zero whenever w is any multiple of 2^h that approaches infinity.

4.3 Values of *i* and *v* which Maximize Correction Term

Now that an upper bound function for the maximum error of any pixel in any given class of lines has been found, the only work that is necessary is to find the maximum value of the function. Obviously, the maximum value of the function in Theorem 5 is found by maximizing $c_{v,\,h,\,i}$.

In this section, the values of i and v which maximize $c_{v,\,h,\,i}$ for any given value of h will be determined by recursively defining $c_{v,\,h,\,i}$. This is done by first defining $D\_ops_{v,\,h,\,i}$ with respect to $n_{0,\,0}$ and then defining $D\_ops_{v,\,h,\,i}$ with respect to the appropriate node at level 1. Finally, the two equations for $D\_ops_{v,\,h,\,i}$ are set equal to each other, and then $c_{v,\,h,\,i}$ is solved for. As noted earlier, it follows from Theorem 4 that

$$D\_ops_{v,\,h,\,i} = \frac{iv - c_{v,\,h,\,i}}{2^h}$$

in all cases. However, there are two cases when defining $D\_ops_{v,\,h,\,i}$ in terms of a SD-subtree rooted at level 1: either i is in the left SD-subtree or i is in the right SD-subtree of the root node. Also, there are two subcases for each case: either v is even, or v is odd. The case where i is in the left SD-subtree will be considered first. For nodes in the left SD-subtree, $D\_ops_{v,\,h,\,i}$ can also be defined as:

$$D\_ops_{v,\,h,\,i} = \frac{i_{h-1}v_{1,\,0} - c_{v_{1,\,0},\,h-1,\,i_{h-1}}}{2^{h-1}}$$

where $i_{h-1} = b^{i}_{h-2}b^{i}_{h-3}\ldots b^{i}_{0}$. Setting the two equations for $D\_ops_{v,\,h,\,i}$ equal to each other and multiplying out the denominators we obtain:

$$iv - c_{v,\,h,\,i} = 2(i_{h-1}v_{1,\,0} - c_{v_{1,\,0},\,h-1,\,i_{h-1}}).$$

Then, if v is even

$$c_{v,\,h,\,i} = 2c_{v_{1,\,0},\,h-1,\,i_{h-1}}. \tag{1}$$

If v is odd

$$iv - c_{v,\,h,\,i} = 2(i_{h-1}\frac{v-1}{2} - c_{v_{1,\,0},\,h-1,\,i_{h-1}}) \text{ and}$$
$$c_{v,\,h,\,i} = i + 2c_{v_{1,\,0},\,h-1,\,i_{h-1}}. \tag{2}$$

Using similar arguments, it can be proven that when i is in the right SD-subtree of the root node and v is even, the value of c equals:

$$c_{v,\,h,\,i} = 2c_{v_{1,\,1},\,h-1,\,i_{h-1}}. \tag{3}$$

When v is odd, then c equals:

$$c_{v,\,h,\,i} = 2^{h} - i + 2c_{v_{1,\,1},\,h-1,\,i_{h-1}}. \tag{4}$$

From (1), (2), (3), and (4), the values of i which maximize c can be determined. Afterwards, induction can be used to determine the values of v which maximize c. Therefore, summarizing these discussions, we have the following theorems.

Theorem 6: If h is an even integer greater than or equal to two, then $c_{v,\,h,\,i}$ is maximized when:

$$b^{i}_{h-1}b^{i}_{h-2}\ldots b^{i}_{0} = (01)^{h/2} \text{ and } b^{v}_{h-1}b^{v}_{h-2}\ldots b^{v}_{0} = (10)^{(h/2)-1}11$$
$$\text{or}$$
$$b^{i}_{h-1}b^{i}_{h-2}\ldots b^{i}_{0} = (10)^{(h/2)-1}11 \text{ and } b^{v}_{h-1}b^{v}_{h-2}\ldots b^{v}_{0} = (01)^{h/2}.$$

Theorem 7: If h is an odd integer greater than or equal to one, then $c_{v,\,h,\,i}$ is maximized when:

$$b^{i}_{h-1}b^{i}_{h-2}\ldots b^{i}_{0} = (01)^{(h-1)/2}1 \text{ and } b^{v}_{h-1}b^{v}_{h-2}\ldots b^{v}_{0} = (01)^{(h-1)/2}1$$
$$\text{or}$$
$$b^{i}_{h-1}b^{i}_{h-2}\ldots b^{i}_{0} = (10)^{(h-1)/2}1 \text{ and } b^{v}_{h-1}b^{v}_{h-2}\ldots b^{v}_{0} = (10)^{(h-1)/2}1.$$

4.4 Express ε in a Convenient Form

Since the values of v and i which maximize c for any given value of h are known, the value of $\max(c_{v,\,h,\,i})$ can be found using the following equation:

$$c_{v,\,h,\,i} = iv - 2^{h}D\_ops_{v,\,h,\,i}.$$

However, it is desirable to find some other way to define $\max(c_{v,\,h,\,i})$ because it is cumbersome to compute $D\_ops_{v,\,h,\,i}$. Alternative definitions are given in the lemmas below. These definitions are then used to obtain formulas for the maximum error of a pixel for the different classes of lines in the theorems which follow.

Lemma 4: If h is an even integer, then

$$\max (c_{v,\,h,\,i}) = \sum_{y\,=\,1,\,3,\,\ldots}^{h\,-\,1} (y + 2)2^{y-1}.$$

Proof: Without loss of generality, assume $n = 10$ and $h = 6$. As noted earlier, the values of $S\_ops_{v,\,h,\,i}$ and $D\_ops_{v,\,h,\,i}$ are found in a similar manner. It is also noted in [6] that there is a great deal of similarity between $D\_ops_{v,\,h,\,i}$ and the value of *iv*. More specifically, the nine leftmost columns of the bits forming *iv* and $2^h D\_ops_{v,\,h,\,i}$ are the same. By Theorem 6, $c_{v,\,h,\,i}$ is maximized when $i = 101011_2$ and the last six bits of v equal 010101_2. Under these conditions, all of the values of inc_y in the summation for $D\_ops_{v,\,h,\,i}$ equal zero. Therefore, $\max(c_{v,\,h,\,i})$ is found by determining the six rightmost columns of the numbers whose sum equals *iv*. The lemma is concluded by mathematical induction on each pair of columns, e.g., the sum of each pair of columns is $2^y(y + 1) / 2 + 2^{y-1}$ or $(y + 2)2^{y-1}$ for $y = 1, 3$, and 5 in the example given here. ☐

Lemma 5: If h is an odd integer, then

$$\max (c_{v,\,h,\,i}) = \sum_{y\,=\,0,\,2,\,\ldots}^{h\,-\,1} (y + 2)2^{y-1}.$$

Proof: The proof is similar to that for Lemma 4. It is omitted for brevity. ☐

Theorem 8: If $\lfloor \log_2 v \rfloor$ is an even integer greater than or equal to two, then

$$\max (|\varepsilon|) < \frac{\displaystyle\sum_{y\,=\,1,\,3,\,\ldots}^{\lfloor \log_2 v \rfloor - 1} (y + 2)2^{y-1}}{2^{\lfloor \log_2 v \rfloor}} + 1.$$

Proof: The theorem follows from Theorem 5, Lemma 4, and the fact that $h = \lfloor \log_2 v \rfloor$. ☐

Theorem 9: If $\lfloor \log_2 v \rfloor$ is an odd integer greater than or equal to one, then

$$\max (|\varepsilon|) < \frac{\displaystyle\sum_{y\,=\,0,\,2,\,\ldots}^{\lfloor \log_2 v \rfloor - 1} (y + 2)2^{y-1}}{2^{\lfloor \log_2 v \rfloor}} + 1.$$

Proof: The theorem follows from Theorem 5, Lemma 5, and the fact that $h = \lfloor \log_2 v \rfloor$. ☐

h	v values	$\max(c_{v,\,k,\,i})$	$\max(\lvert\varepsilon\rvert)$
1	$2 \leq v < 4$	1	1.50
2	$4 \leq v < 8$	3	1.75
3	$8 \leq v < 16$	9	2.13
4	$16 \leq v < 32$	23	2.44
5	$32 \leq v < 64$	57	2.78
6	$64 \leq v < 128$	135	3.11
7	$128 \leq v < 256$	313	3.45
8	$256 \leq v < 512$	711	3.78

Table 2. The range of values of v, the maximum value of the correction term associated with $D\_ops_{v,\,k,\,i}$, and the maximum possible error of a pixel set in an h-complete line for $h = 1, 2, \ldots, 8$.

Using the above formulas, the upper bound for the maximum distance of any pixel set by the RB (and MRB) algorithm from the true line for various classes of lines is given in Table 2. Although these distances are somewhat large, one must keep in mind that the errors are approached only in certain cases for longer lines. These bounds are intended to merely put theoretical limits on the distance that a pixel may lie from the true line and to give some way to compare the error of the RB algorithm with other algorithms which may have some of its advantages.

5 Reducing the Error

Now that the error bounds have been found, it is possible to determine whether variations of the RB algorithm reduce the error. In this section, the *modified recursive bisection reduced error* (MRBRE) algorithm is described. This variation is similar to the MRB algorithm, except the line is shifted to the left a given number of positions. It is easily done as follows. Rather than processing the usual $w_{h,0}$ S operators at node 0 of level h, process ($w_{h,0}$ div 2) S operators. Other than this, both node 0 and the remaining nodes are processed as before. After all the nodes have been processed, perform the [$w_{h,0}$ - ($w_{h,0}$ div 2)] S operations that were previously ignored. Theorem 10, which can be proven by induction (formal proofs are given in [7]), states that when v is a power of two and w is any multiple of v, all pixels have minimum error once shifting is performed. In general, the maximum error is reduced by $mw_{h+1,1}$ for each class of lines. Of course, the lines produced remain highly regular. Therefore, the MRBRE algorithm reduces the error of the previously described algorithms, while adding virtually no additional computation time.

Theorem 10: If $v = 2^h$ and $w = r2^h$, where r is any integer greater than or equal to one, then the maximum distance of any pixel set by the MRBRE algorithm from the true line is less than or equal to 0.5.

6 Final Remarks

New algorithms, the MRB and MRBRE, which improve the performance of an existing fractal line drawing algorithm were presented and analyzed. Our investigation shows that the space requirements of the earlier RB algorithm can be reduced to a constant, while improving the speed and appearance of the lines being drawn. The resulting algorithms can be faster than Bresenham's algorithm, especially when the lines being drawn are at or near horizontal, diagonal, or vertical. In the past, lines have been used in fractal algorithms to draw fractal curves, trees, etc. The work in this paper shows that there are advantages in using a fractal approach to draw lines as well. Perhaps, there are also advantages in developing comparable algorithms to draw other self-similar geometric shapes, such as circles.

7 References

1. J.E. Bresenham, Algorithm for computer control of a digital plotter, *IBM Systems Journal*, **4**(1), 25-30 (1965).
2. J.E. Bresenham, Run length slice algorithm for incremental lines, in *Fundamental Algorithms for Computer Graphics*, R.A. Earnshaw, Ed., pp. 59-104, NATO ASI Series, Springer-Verlag, New York (1985).
3. G. Casciola, Basic concepts to accelerate line algorithms, *Computers & Graphics*, **12**(3/4), 489-502 (1988).
4. C.M.A. Castle and M.L.V. Pitteway, An application of Euclid's algorithm to drawing straight lines. In *Fundamental Algorithms for Computer Graphics*, R.A. Earnshaw, Ed. NATO ASI Series, Springer Verlag, New York, 1985, 135-139.
5. C.M.A. Castle and M.L.V. Pitteway, An efficient structural technique for encoding 'best-fit' straight lines. *Comput. J.* 30 (1987), 168-175.
6. P. Graham, S.S. Iyengar, and S.Q. Zheng, An efficient line drawing algorithm for parallel machines, LSUBR TR# 91-025.
7. P. Graham, S.S. Iyengar, and S.Q. Zheng, Improved recursive bisection line drawing algorithms, LSUBR TR# 91-031.
8. F.S. Hill, Jr., *Computer Graphics*, Macmillan Publishing Co., New York, 430-431 (1990).
9. B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman, San Francisco (1982).
10. J.R. Rankin, Recursive bisection line algorithm, *Computers & Graphics*, **15**(1), 1-8 (1991).
11. X. Wu and J.G. Rokne, Double-step incremental generation of lines and circles. *Comp. Vision Graphics Image Processing* **37**, 331-334 (1987).

Guarding Polyhedral Terrains

Prosenjit Bose* Thomas Shermer† Godfried Toussaint* Binhai Zhu*

1 Introduction

Victor Klee posed the problem of determining the minimum number of guards sufficient to cover the interior of an n-sided art gallery (polygon) in 1973. Chvátal showed that $\lfloor \frac{n}{3} \rfloor$ guards are sufficient and sometimes necessary to cover the interior of an n-sided art gallery using a lengthy combinatorial argument [Chv75]. An elegant and succinct proof was given by Fisk [Fis78]. Although many similiar problems have been studied in computational geometry [O'R87], [She89], [She92], little is known about guarding an object in three dimensions. In this paper, we present some results on guarding the surface of a polyhedral terrain.

A polyhedral terrain is a connected polyhedral surface in 3-D such that the intersection of the terrain with a vertical line is either empty or a point. The problem of guarding a polyhedral terrain was first investigated by deFloriani, et al. [dFP$^+$86]. They showed that finding the minimum number of guards could be done using a set covering algorithm. Cole and Sharir [CS89] showed that the problem was NP-complete.

2 Guarding Polyhedral Terrains

Two points a, b on or above T are said to be *visible* if the line segment $\overline{ab}$ does not intersect any point strictly below T. A set of guards covers a terrain if every point on the terrain is visible from at least one guard in the set. The guarding problem we study is the following: what is the number of vertex (edge) guards that are always sufficient and sometimes necessary to cover any polyhedral terrain? The combinatorial counterparts of these terrain guarding problems can be expressed as guarding problems on the planar triangulated graph derived from the terrain. We prove the following results.

Theorem 2.1 *In the worst case, the guarding problem on a polyhedral terrain is equivalent to the combinatorial guarding problem on the planar triangulated graph derived from the terrain.*

Theorem 2.2 *There exists a terrain on n vertices, for any $n \equiv 3 \bmod 4$ that requires $\lfloor n/2 \rfloor$ vertex guards.*

Theorem 2.3 $\lfloor n/2 \rfloor$ *vertex guards are always sufficient and sometimes necessary to guard the surface of an arbitrary terrain T with n vertices.*

Theorem 2.4 *There is a planar triangulation that needs at least $(4n-8)/13$ edge guards.*

We now present some practical, efficient algorithms for placing the vertex and edge guards. Since establishing the number of vertex guards and the number of edge guards sufficient to cover a terrain made use of the four color theorem, finding a practical efficient algorithm to place the guards seems unlikely unless a deeper understanding of the problem is achieved. To this end, we present practical algorithms for guard placement which approximate the upper bounds.

---

*School of Computer Science, McGill University

†School of Computing Science, Simon Fraser University

Observation 2.1 *Given a five coloring of the vertices of any terrain, any set of three color classes provides a vertex guarding of the terrain since every face of the terrain is a triangle except possibly the outer face (which need not be guarded).*

Based on this observation, a simple linear time algorithm to place $\lfloor \frac{3n}{5} \rfloor$ vertex guards can be obtained by 5-coloring the vertices of the planar triangulation graph in linear time [CNS81] and, among the five colors, choosing three colors that are minimally used. By extending some of the elegant ideas of Everett and Rivera-Campo [ERC92], we develop a linear time algorithm for placing $\lfloor \frac{2n}{5} \rfloor$ edge guards to cover a polyhedral terrain.

Theorem 2.5 *Given a polyhedral terrain on n vertices. $O(n)$ time is sufficient to find a set S of edges to guard the terrain, where $\|S\| \leq \frac{2n}{5}$.*

3 Closing Remarks and Open Problems

The following table summarizes the known results of guarding polyhedral terrains. Reducing the gap between sufficiency and necessity for edge guards, as well as finding practical algorithms that match the known bounds remain open problems.

	Sufficiency	*Necessity*	*Algorithmic Bounds*
Vertex Guards	$\lfloor n/2 \rfloor$	$\lfloor n/2 \rfloor$	$\lfloor 3n/5 \rfloor$
Edge Guards	$\lfloor n/3 \rfloor$	$\lfloor (4n-8)/13 \rfloor$	$\lfloor 2n/5 \rfloor$

References

[Chv75] V. Chvatal. A combinatorial theorem in plane geometry. *J. Comb. Theory Ser. B*, 18:39–41, 1975.

[CNS81] N. Chiba, T. Nishizeki, and N. Saito. A linear 5-coloring algorithm for planar graphs. *Journal of Algorithms*, 1981.

[CS89] R. Cole and M. Sharir. Visibility problems for polyhedral terrains. *Journal of Symbolic Computation*, 7:11–30, 1989.

[dFP+86] L. deFloriani, B. Falcidieno, C. Pienovi, D. Allen, and G. Nagy. A visibility-based model for terrain features. In *Proc. 2nd International Symposium on Spatial Data Handling*, pages 235–250, 1986.

[ERC92] H. Everett and E. Rivera-Campo. n/3 edge guards are sufficient to guard a polyhedral terrain. *Manuscript in preparation*, 1992.

[Fis78] S. Fisk. A short proof of Chvatal's watchman theorem. *J. Combin. Theory Ser. B*, 24:374, 1978.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[She89] T. Shermer. *Visibility properties of polygons.* PhD thesis, School of Computer Science, McGill University, Montreal, Quebec, 1989.

[She92] T. Shermer. Recent results in art galleries. *Proc. IEEE, Special Issue on Comp. Geom. Ed. G.T. Toussaint*, September, 1992.

Optimal Algorithms for Minimizing Total Channel Density in Channel-Placement Problems*

Moazzem Hossain, Naveed A. Sherwani
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008

Si-Qing Zheng
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803

Abstract

Channel density is used as lower bound for estimating channel width for general purpose designs, however, *total channel density* is a measure of choice for computing lower bound of channel for high-performance designs. Total channel density is the sum of the densities over all columns and is a very accurate measure of the total wire length in a given channel. The total density of a given channel can greatly be improved by making modifications to the placement by laterally shifting some of the blocks. Recently, an $\Theta(n)$ time complexity algorithm has been presented for the case when both rows (boundaries) of the channel consist of one block each.

In this paper, we consider the problem of multiple-block channel. We present an $O(n^2)$ time complexity algorithm for a channel with at most three blocks (CRP-3). Then we extend our result for CRP-4 and for CRP-k, the general channel routing problem with k blocks ($k \geq 3$).

1 Introduction

Channel Routing Problem (CRP) is one of the most fundamental problems in VLSI design automation [1, 5, 6, 8, 9, 10]. CRP calls for the detailed routing of wires between two rows of fixed terminals, one on either side of the channel. The objective of the CRP is to find a routing solution with minimum channel width, *i.e.*, find a routing solution with minimum number of tracks. It is well known that the problem is NP-hard [1]. Thus, it is very unlikely that the channel routing problem can be solved in polynomial time. Development of heuristic algorithms for the channel routing problem has received extensive attention (see [7] for summary). One commonly used lower bound for the channel width is called the *channel density*, which is the maximum number of nets that must cross any vertical section of the channel. In practice, channel density is used as lower bound for calculating channel width for ordinary circuits. *Total channel density* is a measure of choice for calculating lower bound of channel for high-performance circuits. Total channel density is the sum of the densities over all columns and is a very accurate measure of the total wire length in a given channel [9].

In CRP, the terminals on each boundary of the channel can be grouped into several *blocks* [6]. The terminals inside the blocks have fixed positions, and the relative ordering of the blocks are completely fixed, *i.e.*, the blocks cannot be swapped. However, the exact

---

*Research supported in part by National Science Foundation Grant USE-9052346 and ACM/SIGDA Scholarship 1991.

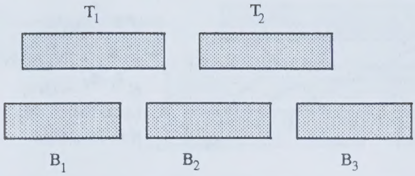


Figure 1: An instance of channel-placement problem

positions of the blocks may not be fixed. All the blocks are allowed to be offset against each other by a different amount without overlapping each other. The problem is to find the relative offset (separation) between each block so that some objective such as channel density, total channel density, channel spread, is minimized. This problem is known as *optimal channel-placement problem* [6]. Figure 1 gives an instance of channel-placement problem.

In [3], a special case of channel-placement problem is considered to minimize the channel density. It has been shown that if each block consists of just one terminal then the problem can be solved optimally in $O(n^2)$ time. In [2], the general problem is considered with the objective of minimizing the channel density and an $O(L^3)$ time complexity algorithm has been proposed, where L is channel length. Tollis [9] presented an $O(n \log n)$ time algorithm for finding a shift of channel boundaries so that the total density of the channel is minimized, assuming that each boundary consists of only one block, *i.e.*, the relative position of all the terminals remain fixed on their respective channel boundaries (we denote this problem as CRP-2). Zheng [11] improved this result and showed that this problem can be solved in $\Theta(n)$ time. In [12], Zheng also showed that the problem of finding minimum total density of a multiterminal-net CRP-2 by lateral shift of channel boundaries can be solved in $\Theta(m)$ time, where m is the total number of terminals on the channel boundaries.

In this paper, we consider the problem of minimizing total channel density. We present an optimal $O(n^2)$ time complexity algorithm for a channel with at most three blocks (CRP-3). Then we present optimal algorithm for CRP-4. We also show that the algorithm can easily be extended to solve CRP-k, general channel with k block, optimally. The significance of the proposed algorithms lies in its application to the area of high-performance IC designs.

2 Problem Formulation

Channel routing problem assumes that the placement of each block to be fixed, hence the boundary of each channel is completely specified and each boundary can be viewed as a single block. The problem of minimizing total channel density is to find a shift S of the top boundary with respect to the bottom boundary so that the total channel density is minimized. This problem has been considered in [10] and a linear time algorithm has been presented.

Since the algorithm presented in [10] serves as a basis of our result, for ease of reference, we briefly describe the algorithm. Initially, it is assumed that the top boundary is placed in such a position that the leftmost terminal of the bottom boundary is immediately to the right of the rightmost terminal of the top boundary. At any shift S of the

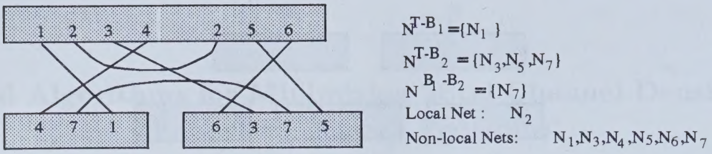


Figure 2: Example of different types of nets

top boundary with respect to the initial position, the span of a net N_i, $I_i(S)$, is defined to be the horizontal length of the net at shift S. Then the total density of the channel at any shift S is $D_t(S) = \sum_{N_i \in \mathcal{N}} I_i(S)$, where $\mathcal{N}$ is the set of nets in the given channel.

A net is called *trivial* if both of its top and bottom terminal lie on the same column. A net N_i is trivial with respect to a shift S if N_i becomes trivial at shift S, *i.e.*, if $I_i(S) = 0$. Following lemma on the trivial net with respect to shift S has been proved in [10].

Lemma 1 [10] *Given a CRP-2, there exist a minimum total density shift S of the top boundary with respect to the bottom boundary, such that at least one net is trivial with respect to S.*

Following algorithm to find the minimum total density shift is based on this lemma [10].

```
Algorithm MIN-TOTAL-DENSITY
1.    Compute $I_i(0)$, for $1 \le i \le n$;
2.    Let $k = \lfloor \frac{n+1}{2} \rfloor$;
3.    Let $S = k$-th smallest number in $\{I_i(0) | 1 \le i \le n\}$;
4.    return (S);
end MIN-TOTAL-DENSITY
```

The more general problem assumes that each boundary consists of several blocks. Let $T_1, T_2, \ldots, T_p$, and $B_1, B_2, \ldots, B_q$ are the blocks on the top and bottom boundary respectively in a given CRP. The relative ordering of each block on each boundary is fixed, *i.e.*, T_i must be in the left of T_{i+1}, for $1 \le i \le p-1$, and B_j must be in the left of B_{j+1}, for $1 \le j \le q-1$. However, the exact position of each block is not fixed. The CRP-k problem can be stated as: *Given a channel with k blocks such that $T_1, T_2, \ldots, T_p$, and $B_1, B_2, \ldots, B_q$ are the blocks on the top and bottom boundary respectively, and $k = p + q$. The relative ordering of each block on each boundary is fixed. Find the position for each block by laterally shifting each block so that the total density of the channel is minimized.*

In this paper, we restrict ourselves to two-terminal nets. A net is called *local net* if its two terminals are on the same block, otherwise the net is called *non-local net*. Since for any local net, its contribution to the total density is constant over all shifts, we only need to consider non-local nets. We denote a net N_i to be in set $\mathcal{N}^{b_1-b_2}$ iff its terminals are on blocks b_1 and b_2, for $b_1, b_2 \in \{T_1, T_2, \ldots, T_p, B_1, B_2, \ldots, B_q\}$, and $b_1 \neq b_2$. Note that for a channel consisting of k blocks, the set of nets can be partitioned into $k(k-1)/2$ different sets of non-local nets. For instance, a CRP consisting of one top block, say T, and two bottom blocks, B_1 and B_2, there are three sets of non-local nets, $\mathcal{N}^{T-B_1}, \mathcal{N}^{T-B_2}$, and $\mathcal{N}^{B_1-B_2}$, as shown in Figure 2.

Initially, we assume that $T_1, T_2, \ldots, T_p, B_1, B_2, \ldots, B_q$ are positioned tightly together (as shown in Figure 3), that is, there is no gap between any two blocks. We call this

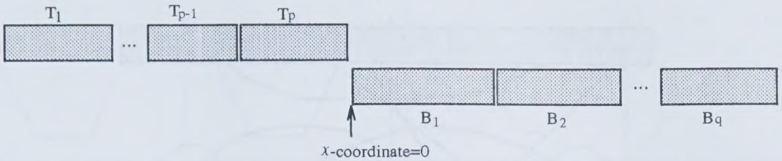


Figure 3: Initial position

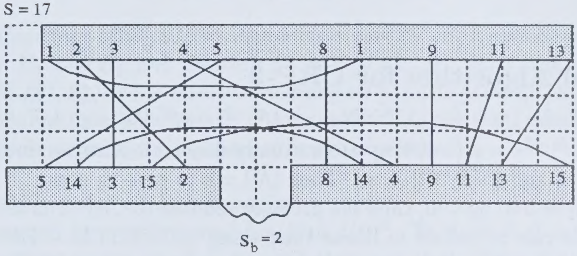


Figure 4: Minimum total density CRP-3

position *initial position.* Note that the terminals can be viewed as positioned along the x-axis and can be referred by their x-coordinates. Let $N_i \in \mathcal{N}^{b_1-b_2}$, for $b_1, b_2 \in \{T_1, T_2, \ldots, T_p, B_1, B_2, \ldots, B_q\}$, and $b_1 \neq b_2$, then we refer the x-coordinate of its two terminals as $x(b_1, N_i)$, and $x(b_2, N_i)$. If b_1 and b_2 both are either top or bottom blocks then $x(b_1, N_i) < x(b_2, N_i)$. We assume that the x-coordinate of the leftmost (rightmost) terminal of block B_1 (T_p) is 0.

Without loss of generality, we assume that B_1 is stationary. The problem of finding the positions for each block is equivalent to finding separations between each adjacent blocks in the same boundary, and a shift S of T_1 with respect to the initial position so as to minimize the total channel density. For p blocks on the top boundary, we have to find $p-1$ different separations, S_{t_1}, S_{t_2}, …, $S_{t_{p-1}}$, such that S_{t_1} is the separation between T_1 and T_2, S_{t_2} is the separation between T_2 and T_3 and so on. Similarly, for q blocks on the bottom boundary, we have to determine $q-1$ different separations, S_{b_1}, S_{b_2}, …, $S_{b_{q-1}}$. Figure 4 gives an example of a CRP-3 with minimum total density shift S and the separation S_b.

In development of our algorithm, we use the notion of *multiset* along with the notion of set. Just like a set, a multiset is a collection of elements, however, unlike a set, a multiset is permitted to have multiple occurrences of the same element. We need to define the multiset operations. Let a_e, and b_e be the number of occurrences of element e in multisets A and B, respectively. The operation union of two multisets A and B results in a multiset C in which the occurrences of e is $a_e + b_e$. The operation intersection of two multisets A and B results in a multiset C in which the occurrences of e is $\min\{a_e, b_e\}$. Let $C = A \cap B$, and c_e be the number of occurrences of element e in C. Then the operation $A - B$ results in a multiset D in which the number of occurrences of e is $a_e + c_e$.

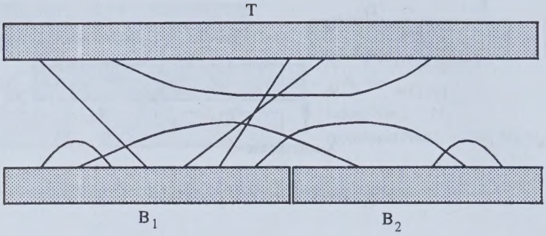


Figure 5: Example for $n_2 = 0$

3 Optimal Algorithm for CRP-3

Let, $\mathcal{N} = \mathcal{N}^{T-B_1} \cup \mathcal{N}^{T-B_2} \cup \mathcal{N}^{B_1-B_2}$, and $\mathcal{N}^{T-B_1} \cap \mathcal{N}^{T-B_2} = \phi$, $\mathcal{N}^{T-B_1} \cap \mathcal{N}^{B_1-B_2} = \phi$, and $\mathcal{N}^{T-B_2} \cap \mathcal{N}^{B_1-B_2} = \phi$ (i.e., they are mutually disjoint). Also assume that $|\mathcal{N}^{T-B_1}| = n_1$, $|\mathcal{N}^{T-B_2}| = n_2$ and $|\mathcal{N}^{B_1-B_2}| = n_3$. Then $|\mathcal{N}| = n = n_1 + n_2 + n_3$.

Clearly, if $n_1 = 0$ or $n_2 = 0$, then the problem reduces to CRP-2 as shown in Figure 5. As a result, this can be solved in linear time using MIN-TOTAL-DENSITY. Thus, we assume that $n_1 \neq 0$ and $n_2 \neq 0$. Note that $(B_1 - B_2)$ nets (nets in $\mathcal{N}^{B_1-B_2}$) will contribute more to the total density if blocks B_1 and B_2 are not placed close next to each other. However, if B_1 and B_2 are placed at a distance from each other, the $(T - B_1)$ and/or $(T - B_2)$ nets may contribute less total density for some shifts, which may lead to a better solution.

With respect to the initial position of T, B_1 and B_2, we define the following parameters,

$I_i^{T-B_1}(0) = x(B_1, N_i) - x(T, N_i)$, for all $N_i \in \mathcal{N}^{T-B_1}$.

$I_i^{T-B_2}(0, S_b) = x(B_2, N_i) - x(T, N_i)$, for all $N_i \in \mathcal{N}^{T-B_2}$.

A straight forward algorithm to compute S and S_b that produce minimum total density is presented below.

```
Algorithm MINSHIFT1
1.      Compute $I_i^{T-B_1}(0)$ for each net $N_i \in \mathcal{N}^{T-B_1}$;
2.      Compute $I_j^{T-B_2}(0,0)$ for each net $N_j \in \mathcal{N}^{T-B_2}$;
3.      Let $D^* = \sum_{N_i \in \mathcal{N}^{T-B_1}} I_i^{T-B_1}(0) + \sum_{N_j \in \mathcal{N}^{T-B_2}} I_j^{T-B_2}(0) +$
        total span of all nets in $\mathcal{N}^{B_1-B_2}$ at shift $S_b = 0$;
4.      Let $S_b^* = 0$;
5.      For all candidate shift $S_b$ do
5.1.        Compute $I_j^{T-B_2}(0, S_b)$ for each net $N_j \in \mathcal{N}^{T-B_2}$;
5.2.        Let $L$ = Total span of all nets in $\mathcal{N}^{B_1-B_2}$ at shift $S_b$;
5.3.        $S$ = MIN-TOTAL-DENSITY;
5.4.        Let $D$ = total density of nets of $\mathcal{N}^{T-B_1}$ and $\mathcal{N}^{T-B_2}$ w.r.t. $S$ and $S_b$;
5.5.        $D = D + L$;
5.6.        if ($D < D^*$) then
5.6.1.          $D^* = D$; $S^* = S$; $S_b^* = S_b$;
6.      return ($S^*, S_b^*$);
end MINSHIFT1
```

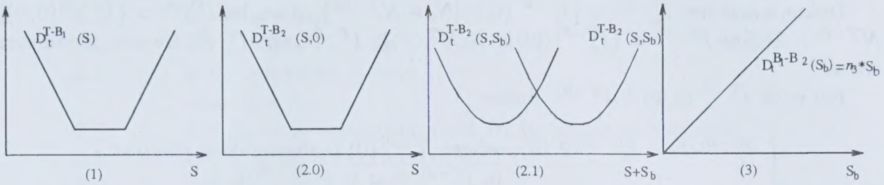


Figure 6: Behavior of different density functions

Theorem 1 *Algorithm MINSHIFT1 computes a pair (S^*, S_b^*), which specify the separation S_b^* between B_1 and B_2 and the shift S^* of T w.r.t. B_1, such that, the total density corresponding to (S^*, S_b^*) is the minimum among all possible pairs (S', S_b'). Furthermore, the time complexity of the algorithm is $O(pn)$, where n is the total number of nets and p is the number of S_b separations considered in the algorithm.*

Clearly, p, the number of S_b separations considered in the algorithm MINSHIFT1, may not be bounded by a polynomial function $f(n)$[1]. To show that a polynomial time algorithm is possible, we have to find a p such that $p \leq f(n)$, where $f(n)$ is a polynomial of n, the total number of nets.

Note that p is less than or equal to the length of T. To limit the number of S_b to be considered, we observe the following simple facts:

1. Since B_1 is fixed, $I_i^{T-B_1}(0)$, for all nets $N_i^{T-B_1} \in \mathcal{N}^{T-B_1}$, is computed once and used for all possible S_b.
2. The contribution of nets in $\mathcal{N}^{B_1-B_2}$ to the total density increase $n_3 * S_b$, compared with $S_b = 0$.
3. For a particular S_b, the overall effect of all nets to the total density may decrease.

For any (T, B_1, B_2) instance, the total density D_t is a function of two variables, S, and S_b. We can thus view the problem of computation of $D_t(S, S_b)$ as three subproblems: (1) computation of $D_t^{T-B_1}(S, S_b)$, (2) computation of $D_t^{T-B_2}(S, S_b)$, and (3) computation of $D_t^{B_1-B_2}(S, S_b)$, one for each type of non-local nets.

Since $D_t^{T-B_1}(S, S_b)$ is not related to S_b, we write it as $D_t^{T-B_1}(S)$, similarly, $D_t^{B_1-B_2}(S, S_b)$ is not related to S, and we write it as $D_t^{B_1-B_2}(S_b)$. Figure 6 shows the behavior of these functions.

Our objective is to find S and S_b, such that $D_t(S, S_b) = D_t^{T-B_1}(S) + D_t^{T-B_2}(S, S_b) + D_t^{B_1-B_2}(S_b)$ is minimum.

Since functions (1) and (2.0) (Figure 6) are fixed, we need to consider the contribution of functions (2.1) and (3) (Figure 6) to $D_t(S, S_b)$ as S_b changes.

---

[1] In standard cell design, each cell can be viewed as a block. Examination of benchmarks and industrial designs reveals that most standard cell designs have 50% to 85% "vacant" terminals depending on the given boundary. A terminal is said to be vacant if it is not required for any net connection. The large number of vacant terminals in standard cell designs is due to the fact that each logical terminal (input and output) is provided on both sides of the block, but, in most cases, need only be connected on one side.

Define a multiset $I_m^{T-B_2} = \{I_i^{T-B_2}(0,0) | N_i \in \mathcal{N}^{T-B_2}\}$. Also, let $I^{T-B_2} = \{I_i^{T-B_2}(0,0) | N_i \in \mathcal{N}^{T-B_2}\}$. Define $I^{T-B_1} = \{I_j^{T-B_1}(0) | N_j \in \mathcal{N}^{T-B_1}\}$. I^{T-B_1}, and I^{T-B_2} are sets of common sense.

For each $I_i^{T-B_2}(0,0) \in I^{T-B_2}$, define

$$r_i = \begin{cases} I_j^{T-B_1}(0) - I_i^{T-B_2}(0,0), & \text{where } I_j^{T-B_1}(0) \text{ is the smallest element} \\ & \text{in } I^{T-B_1} \text{ that is} \geq I_i^{T-B_2}(0,0). \\ \infty, & \text{if the above defined } I_j^{T-B_1}(0) \text{ does not exist.} \end{cases}$$

Suppose, $|I^{T-B_2}| = k_1$, then we have $r_1, r_2, \ldots, r_{k_1}$.

Without loss of generality, assume that $r_1 \leq r_2 \leq \ldots \leq r_{k_1}$. Then, we can obtain a sequence of k pairs: $[(r_1, m_1), (r_2, m_2), \ldots, (r_{k_1}, m_{k_1})]$, where, m_j is the multiplicity of $I_j^{T-B_2}(0,0) \in I^{T-B_2}$ in $I_m^{T-B_2}$ (*i.e.*, if $I^{T-B_2}(0,0) \in I^{T-B_2}$ appears z times in $I_m^{T-B_2}$, then $m_j = z$).

Let $r_0 = 0$ and $m_0 = 0$. In the following lemma, we show how to compute S_b for each r_i, for $1 \leq i \leq k_1$.

Lemma 2 *For each* r_i, $1 \leq i \leq k_1$, $r_{i-1} \leq S_b \leq r_i$ *such that* $y = S_b(n_3 - m_i) + \sum_{l=0}^{i-1}(S_b - r_l)m_l$ *is minimum.*

Calculating S_b such that y is minimized is the most critical part in lemma 2. Let us examine y closely.

$$\begin{aligned} y &= S_b(n_3 - m_i) + \sum_{l=0}^{i-1}(S_b - r_l)m_l \\ &= S_b(n_3 - m_i + \sum_{l=0}^{i-1} m_l) - \sum_{l=0}^{i-1} r_l m_l \end{aligned}$$

Let $C_1 = n_3 - m_i + \sum_{l=0}^{i-1} m_l$ and $C_2 = \sum_{l=0}^{i-1} r_l m_l$.

Then, we want to minimize $y = S_b C_1 - C_2$, for some S_b. Since, r_i's are sorted in ascending order, following lemma shows how to calculate S_b.

Lemma 3 *For each* r_i, $1 \leq i \leq k_1$,

$$S_b = \begin{cases} r_{i-1} & \textit{if } C_1 \geq 0, \textit{ where } C_1 = n_3 - m_i + \sum_{l=0}^{i-1} m_l \\ r_i & \textit{otherwise} \end{cases}$$

Based on lemma 2 and lemma 3, we develop an algorithm (MINSHIFT2) that finds S_b for each r_i, $1 \leq i \leq k_1$, and use S_b to adjust the block B_2 to obtain an instance, then apply algorithm MIN-TOTAL-SHIFT to find S^*, such that the total density $D_t^i(S^*, S_b)$ is minimum. The solution is a pair of (S^*, S_b), for which the total density is minimum among all $D_t^i(S^*, S_b)$. The detailed algorithm is presented below:

Algorithm MINSHIFT2

```
1.  Compute $n_1$, $n_2$, and $n_3$;
2.  Compute $[(r_1, m_1), (r_2, m_2), \ldots, (r_{k_1}, m_{k_1})]$;
3.  Let $r_0 = 0$, $m_0 = 0$;
4.  Let $D_t^i = 0$, for $1 \leq i \leq k_1$;
5.  for $i = 1$ to $k_1$ do
```

```
5.1.        $C_1 = n_3 - m_i$;
5.2.        for $j := 0$ to $i - 1$ do $C_1 = C_1 + m_j$;
5.3.        if $C_1 \geq 0$ then $S_b = r_{i-1}$;
            else $S_b = r_i$;
5.4.        Fix $B_2$ at $S_b$ separation from $B_1$ to obtain an instance;
5.5.        Let $S$ = MIN-TOTAL-DENSITY;
5.6.        Let $D_t^i = D_t(S, S_b)$;
6.      for all $D_t^i$, find the $(S, S_b)$ corresponding to the smallest total density;
7.      return $(S, S_b)$;
end MINSHIFT2
```

Step 1 of the algorithm can be computed $O(n)$ time. The time complexity of step 2 is $O((n_1 + n_2)\log(n_1 + n_2))$. The time complexity of steps 5.2 is $O(k_1)$, and the time complexity of steps 5.4 and 5.5 is $O(n_1 + n_2 + n_3)$. Therefore the time complexity of step 5 is $O(k_1^2 + k_1(n_1 + n_2 + n_3))$. The time complexity of step 6 is $O(k_1)$. Since $k_1 \leq n_2$, the total time complexity of the algorithm is

$$\begin{aligned} T(n) &= T(n_1 + n_2 + n_3) \\ &= O((n_1 + n_2)\log(n_1 + n_2) + n_2(n_1 + n_2 + n_3)) \\ &\leq O(n^2) \end{aligned}$$

Theorem 2 *Given an instance of CRP-3, algorithm MINSHIFT2 computes the minimum total density shift (S, S_b) in $O(n^2)$ time.*

Proof: The details of the proof is omitted and may be found in [4] ■

4 Optimal Algorithm for CRP-4

In this section, we extend our result for CRP-3 to the case when both top and bottom boundary consist of two blocks each, T_1, T_2, and B_1, B_2 respectively (we denote this problem as CRP-4). The problem is to find two separations S_t, and S_b, between T_1, T_2 and B_1, B_2 respectively and a shift S of T_1 with respect to the initial position so that the total density is minimized. Therefore, the total density function $D_t(S, S_t, S_b)$ is function of three variables.

Note that non-local nets can be grouped into six sets: $\mathcal{N}^{T_1-B_1}$, $\mathcal{N}^{T_1-B_2}$, $\mathcal{N}^{T_2-B_1}$, $\mathcal{N}^{T_2-B_2}$, $\mathcal{N}^{T_1-T_2}$, $\mathcal{N}^{B_1-B_2}$. As a result, we can divide the problem into six subproblem: $D_t^{T_1-B_1}(S, S_t, S_b)$, $D_t^{T_1-B_2}(S, S_t, S_b)$, $D_t^{T_2-B_1}(S, S_t, S_b)$, $D_t^{T_2-B_2}(S, S_t, S_b)$, $D_t^{T_1-T_2}(S, S_t, S_b)$, $D_t^{B_1-B_2}(S, S_t, S_b)$.

We assume that $|\mathcal{N}^{T_1-B_1}| = n_{T_1-B_1}$, $|\mathcal{N}^{T_1-B_2}| = n_{T_1-B_2}$, $|\mathcal{N}^{T_2-B_1}| = n_{T_2-B_1}$, $|\mathcal{N}^{T_2-B_2}| = n_{T_2-B_2}$, $|\mathcal{N}^{T_1-T_2}| = n_{T_1-T_2}$, $|\mathcal{N}^{B_1-B_2}| = n_{B_1-B_2}$. It is easy to see that for any fixed S_t or S_b, the problem reduces to CRP-3 and can be solved in $O(n^2)$ time.

The basic idea of the algorithm is to compute all possible S_t separations for any particular S_b separation. The S_b separations are computed with respect to the initial position, *i.e.*, considering S_t separation 0. Since, S_t separation is temporarily fixed to zero, we can effectively compute all possible S_b separations using lemma 2 and lemma 3.

We now describe in detail how to compute all possible S_t separations for a given S_b separation. With respect to a particular S_b separation with $S = 0$ and $S_t = 0$, we define the following:

$$I^{T_1-B} = \{I_j^{T_1-B_1}(0, 0, S_b) | 1 \leq j \leq n_{T_1-B_1}\} \cup \{I_j^{T_1-B_2}(0, 0, S_b) | 1 \leq j \leq n_{T_1-B_2}\} \cup \{0\}$$

For all nets in $\mathcal{N}^{T_2-B_1}$ and $\mathcal{N}^{T_2-B_2}$, we define a multiset

$$I_m^{T_2-B} = \{I_j^{T_2-B_1}(0,0,S_b) | 1 \le j \le n_{T_2-B_1}\} \cup \{I_j^{T_2-B_2}(0,0,S_b) | 1 \le j \le n_{T_2-B_2}\}$$

Also define

$$I^{T_2-B} = \{I_j^{T_2-B_1}(0,0,S_b) | 1 \le j \le n_{T_2-B_1}\} \cup \{I_j^{T_2-B_2}(0,0,S_b) | 1 \le j \le n_{T_2-B_2}\}$$

For each $I_i^{T_2-B} \in I^{T_2-B}$, we define

$$r_i^t = \{I_i^{T_2-B} - I_j^{T_1-B} | I_j^{T_1-B} \in I^{T_1-B} \text{ is the largest such that } I_j^{T_1-B} \le I_i^{T_2-B}\}$$

From these r_i^t values, using lemma 2 and lemma 3 of the previous section, we can compute all possible S_t separations for a particular S_b separation. We now formally present the algorithm for CRP-4.

```
Algorithm MINSHIFT3
Compute $[(r_1^b, m_1^b), (r_2^b, m_2^b), \ldots, (r_{k_1}^b, m_{k_1}^b)]$;
Let $r_0^b = 0$, $m_0^b = 0$;
$D_t^* = \infty$;
for $i = 1$ to $k_1$ do
    Compute $S_b$ from $r_{i-1}^b$ and $r_i^b$;
    use $S_b$ to adjust the block $B_2$ to obtain an instance;
    Compute $[(r_1^t, m_1^t), (r_2^t, m_2^t), \ldots, (r_{k_2}^t, m_{k_2}^t)]$;
    Let $r_0^t = 0$, $m_0^t = 0$;
    for $j = 1$ to $k_2$ do
        Compute $S_t$;
        use $S_t$ to adjust the block $T_2$ to obtain an instance;
        Let $S$ = MIN-TOTAL-DENSITY;
        if $(D_t(S, S_t, S_b) < D_t^*)$ then
            $D_t^* = D_t(S, S_t, S_b)$;
            $S^* = S$; $S_t^* = S_t$; $S_b^* = S_b$;
return $(S^*, S_t^*, S_b^*)$;
end MINSHIFT3
```

Theorem 3 *Given an instance of CRP-4, where both top and bottom boundary consist of two block, algorithm MINSHIFT3 computes the minimum total density shift* (S, S_t^*, S_b^*) *in* $O(n^3)$ *time.*

Proof: The proof of this theorem follows the same line of argument presented for theorem 2 with more cases and we omit the proof for the sake of brevity. ■

It is easy to see that the algorithm presented above can easily be extended for general channel with more than four blocks (k blocks, $k > 3$). We omit the details of the extension and may be found in [4].

Theorem 4 *Given an instance of CRP-k, for* $k \ge 3$ *algorithm MINSHIFT4 computes the minimum total density shift* $(S, S_{t_1}, S_{t_2}, \ldots, S_{t_{p-1}}, S_{b_1}, S_{b_2}, \ldots, S_{b_{q-1}})$ *in* $O(n^{k-1})$ *time, where* n *is the total number of nets in the channel and* p, *and* q *are the total number of blocks on the top and bottom boundary respectively and* $k = p + q$.

5 Conclusion

In this paper, we have presented algorithms for optimal placement of blocks on the boundaries of a given channel so that the total channel density is minimized. We have presented an $O(n^2)$ time complexity algorithm for CRP-3. Then we have extended our result for CRP-4 and at last for CRP-k, the general block channel problem with k blocks ($k \geq 3$). Currently we are working on the problem minimizing the total channel density in multiple-channel environment.

References

[1] D. J. Brown and R. L. Rivest, "New Lower Bounds on Channel Width," *Proc. CMU Conf. on VLSI Systems and Computations*, 1981.

[2] Y. Cai and D. F. Wong, "Optimal Channel Pin Assignment", *IEEE Trans. on CAD*, vol. 10, no. 11, pp. 1413-1423, Nov. 1991.

[3] I. S. Gopal, D. Coppersmith, and C. K. Wong, "Optimal Wiring of Movable Terminals," *IEEE Transactions on Computers*, C-32(9), pp. 845-858, 1983.

[4] M. Hossain, N. A. Sherwani, and S. -Q. Zheng, "Optimal Algorithms for Minimizing Total Channel Density in Channel-Placement Problems", technical report no. TR/92-02, Department of Computer Science, Western Michigan University, 1992.

[5] A. S. LaPaugh and R. Y. Pinter, "On Minimizing Channel Density by Lateral Shifting," Proc. of ICCAD'83 pp. 223-234, 1983.

[6] T. Lengauer, "Combinatorial Algorithms for Integrated Circuit Layout," John Wiley & Sons, 1990.

[7] B. Preas and M. Lorenzetti, "Physical Design Automation of VLSI Systems," The Benjamin/Cummings Publishing Company, Inc., 1988.

[8] R. Tamassia and I. G. Tollis, "On Improving Channel Routability by Lateral Shifting of the Shores," *SIGDA Newsletter*, vol. 18, No. 1, pp. 18-30, 1988.

[9] I. G. Tollis, "On Minimizing the Total-Density of a Channel," *Proceedings of 19th Southeastern International Conf. on Combinatorics, Graph Theory and Computing*, 1988.

[10] I. G. Tollis and S.-Q. Zheng, "Ensuring Compact VLSI Layout by Adjusting Channel Placement", to appear in the *International Journal of Computer and Software Engineering*.

[11] S. -Q. Zheng, "A Linear-Time Algorithm for Minimizing the Total Density of a Channel," Technical Report #88-026, Department of Computer Science, LSU, 1988.

[12] S. -Q. Zheng, "Minimizing the Total Density of a Channel in Optimal Time," to appear in *Congressus Numerantium*.

ALGORITHMS FOR CHANNEL ROUTING WITH INTERCHANGEABLE TERMINALS

Spyros Tragoudas
Department of Computer Science
Southern Illinois University, Carbondale, IL 62901

Abstract The use of programmable logic cells in VLSI design allows the terminals on these cells to be interchanged since their geometries are programmable. Moreover, the terminals of any logic gate can be interchanged without affecting the functionality of the input circuit. Algorithms for various objective functions, NP-complete results, and heuristics that minimize the channel width in this new channel routing problem are given in [15, 21, 3, 4, 8, 19, 9, 14, 17, 2].

In this paper we consider channels with exits. Let m, D be the number of terminals in the channel, and the maximum number of terminals on a net, respectively. We present an $O(m)$ algorithm that obtains optimal density for channels with exits that have one cell on each side. The algorithm in [4] can only guarantee an approximate density for this input instance. We also show that if one of the two cells has fixed terminals then the density minimization problem is NP-hard. For instances with an arbitrary number of cells we also present: an $\mathrm{O}(m)$ time algorithm for the via minimization problem and an $\mathrm{O}(m^2 \cdot D)$ algorithm for the problem of finding a maximum planar subset of nets in a channel.

1 Introduction

Channel routing is a fundamental problem in VLSI design. In a channel routing problem the input consists of a collection of *cells* placed on adjacent positions and partitioned in two groups called the *top* and the bottom channel side, respectively. A rectilinear channel is formed between the two channel sides. Each cell contains *terminals* (otherwise called pins). Terminals are labeled according to the net they belong. Terminals labeled with zero do not belong to any net and we call them *zero terminals.*

Algorithms to route channels with fixed terminals have been proposed by several authors [11, 16]. Recently, a new model for channel routing has been proposed [9] in which we are considering that some or all of the cells on the channel sides consist of terminals whose relative positions within the cell is not fixed, i.e., the terminals can be interchanged in their cells. This is allowed if the cells have been implemented with programmable logic since their geometries are programmable. Programmable logic cells (i.e, ROM's and PLA's) are widely used in VLSI design because of their structural regularity and flexibility [9]. Terminals can be interchanged only if they are in the same cell. A methodology for performing the routing in this type of channels consists of two steps: We first interchange terminals within each cell in order to minimize some routing requirements (e.g., *density*, *number of layers*, *number of vias* etc.), and second, route the "new" channel.

Efficient heuristics that result in minimum number of tracks are given in [9, 8, 17, 14]. Lin and Sahni [15] present a linear time algorithm that obtains *maximum terminal alignment.* We say that two terminals are aligned if they are on the same net, on opposite channel sides and on the same vertical column of the underlying rectilinear grid. NP-complete results are presented in [19]. In [21] Tollis and Tragoudas present a linear time algorithm to determine whether the channel is river routable, they showed that the problem of minimizing the density is NP-hard in general, but they presented polynomial time algorithms for the special case of channels with two terminal nets and channels with at most one terminal per cell for each net. Leong [12] presents polynomial time algorithms that minimize the density in various restricted channel instances.

Cong [3], and Cong and Khoo [4] examine the density minimization problem on a simpler terminal permutation problem where the terminals are interchangeable anywhere on the channel side they reside, i.e., there is one cell per channel side. The algorithms in [3, 4] handle channels with exits, i.e., there exist external nets. A *left exit* or *left* *external net* is a net that connects at least one terminal outside the channel, the global routing phase has determined that the net will exit the channel from its left end only. We define similarly *right exits (right external nets)* and *external nets with two exits.* Let m be the number of terminals in the channel. For simplicity, assume that the number of external nets is $O(m)$. (This is not true, however, if the number of external nets with two exits and no terminals on the channel is large. In those cases the time complexity will depend on the number of the latter nets as well.) The algorithm in [3] finds an optimal density for channels with two terminal nets in $O(m)$ time. However, the algorithm in [4] guarantees density within one unit form the optimal for instances with multiterminal nets in $O(m \cdot \log m)$ time. Leong [12] also examined the simplified instance in [3, 4] but he only considered channels without exits.

Let T and B be the number of cells at the top and bottom sides of the channel. Throughout this paper we assume that on each channel side the cells are labeled from left to right as c_{T1}, ..., c_{TT} and $c_{B1}, ..., c_{BB}$. Let n, m, e, D be the number of terminal positions on either side of the channel, the total number of the terminals which are endpoints of a net, the number of external nets, and the maximum number of terminals in a net, respectively. We assume that $e = O(m)$. We call *internal* any net which is not external.

In Section 2 we consider Cong and Khoo's [4] channel routing problem where there exists only one cell per side and we present optimal algorithms that minimize the density in $O(m)$ time. There are three definitions of the density in the literature. The first density definition is given in [12, 14, 2, 21]. The *density d* of the channel is defined to be equal to the maximum density at a *column* τ_i of the underlying rectilinear grid; The density at a column τ_i is equal to the number of nets crossing column τ_i, i.e., the number of nets with leftmost terminal to the left or on column τ_i, rightmost terminal to the right or on column τ_i, and at least one terminal outside column τ_i. In [11], the *open* and *closed* density are formally defined. In this paper, we consider only the first density definition. In [22] we show how the presented algorithms can be modified to minimize open and closed density. Moreover, we consider the variation proposed in [4] where only one of the two cells has interchangeable terminals, and we show that the density minimization problem is NP-hard. (In [22] we extend this result to open and close density, we show that special cases of the problem are polynomially solvable.)

In Section 3, we consider the via minimization problem [11, 16] and the problem of finding the maximum number of nets that can be routed in one layer [18]. The problems are studied for the general instance with an arbitrary number of cells with interchangeable terminals on both sides. We present an $O(m)$ time algorithm for the first problem and an $O(m^2 \cdot D)$ time algorithm for the second. (In [22], we also give an $O(m)$ algorithm for the problem of determining whether the channel adopts internal-external layout [11, 23]. We also give an alternative $O(m)$ time algorithm for the maximum planar subset problem on channels that have one cell per side.)

In the literature [24, 7, 5, 20, 5, 13, 2] many algorithms and heuristics have been proposed for a similar but different channel routing problem with movable terminals. Now the terminals are movable but their relative order remains unchanged in their cells. The algorithms and heuristics for this problem do not apply in our channel routing problem with interchangeable terminals. Similar channel routing problems are also discussed in [1, 10].

2 Density Minimization

2.1 One Cell per Channel Side

We initially give a simple algorithm, we call it Algorithm 1, that obtains optimal density if there are no external nets. A similar algorithm has been presented in [12]. We are based on the observation that for this special case the terminals can be interchanged so that the density is either one or two. Let k_i denote the *value* of net N_i, i.e., the difference between the number of terminals on the top side minus the number of terminals on the bottom side. (Therefore the value of a net can be a negative number.) Algorithm 1, uses as a subroutine *Internal_Nets* which interchanges the terminals so that the density is never more than two.

Procedure *Internal_Nets* has as input parameters three stacks $S1$, $S2$ and $S3$ of nets. The stacks are generated from a procedure, which we call the *preprocessing_step*(S_1, S_2, S_3), as follows: First we sort (using bucketsorting) all nets according to their values. Then we scan the sorted list and we push each net in stack $S1$ or $S2$ or $S3$, depending on whether its value are positive, zero or negative, respectively. (The top of each stack has net N_i with the smallest absolute value of k_i among the remaining nets in the stack.)

procedure *Internal_Nets*$(S1, S2, S3)$ **Input:** *A channel with internal nets only.* **Output:** *An interchange of the terminals so that the density (as defined in [2, 12, 14, 21]) is no more than two.*

```
(a) Place all the nets of $S2$ at the leftmost available terminal positions so that they are aligned;
(b) $S = S1$;
(c) while both $S1$ and $S3$ are not empty do  {
        if $S = S1$ then pop a net $N_j$ from $S$ and place it on the leftmost positions as illustrated
            in either Figure 1(a) or 1(d) or 1(e); (Only one of these placement rules can apply.)
        else pop a net $N_j$ from $S$ and place it on the leftmost positions as illustrated in
            either Figure 1(b) or 1(c); (Only one of these two placement rules can apply.)
    (d) if the number of placed terminals on the bottom channel side is less than the number
            of placed terminals on the bottom side then $S = S3$ else $S = S1$.   }
(d) Let $S$ be the nonempty stack.
    Place the terminals of each net $N_j \in S$ at consecutive channel positions so that no two
    nets $N_j, N_k \in S$ intersect each other. (Such placement is always feasible.)
end of Internal_Nets
```

Clearly, the channel density is no more than two. At Step (a), we insist that the nets in S_2 are aligned. This is important for Algorithm 2 (described later) that handles external nets. Let m be the number of the terminals in the channel. The sorting of the nets in the preprocessing step is done with bucketsorting. The buckets correspond to the values of the nets. Bucketsorting takes $O(m)$ time since for every net N_i we have that $k_i \in [-(m-1)/2, (m-1)/2]$. Steps (a)-(d) of *Internal_Nets* require $O(m)$ time. Therefore the overall time complexity is $O(m)$ and the number of nets is $O(m)$.

Algorithm 1

Input: *A channel with one cell per side and internal nets only.* **Output:** *A placement of the terminals so that optimal density (as defined in [2, 12, 14, 21]) is obtained.*

```
(a) Place all the nets $N_i$ with $k_i = 0$ at the leftmost channel positions so that they are aligned;
(b) Transform each net $N_i$ of value $k_i \geq 1$ to the pattern of Figure 2(a) by adding zero terminals
        on the bottom side; (These terminals will be part of $N_i$.)
    if $\not\exists$ enough zero terminals then go to Step (e);
(c) Transform each net $N_i$ of value $k_i \leq -1$ to the pattern of Figure 2(b) by adding zero
        terminals on the top side. (These terminals will be part of $N_i$.)
    if $\not\exists$ enough zero terminals then go to Step (e).
```

```
(d) (Now the density is one.) Place the transformed nets in Steps (b) and (c) at the leftmost
    available positions; exit;

(e) (Now the density is two.) call preprocessing_step($S_1, S_2, S_3$);
    call Internal_Nets($S1, S2, S3$)

end of Algorithm 1
```

Algorithm 1 obtains optimal density in $O(m)$ time. For channels that do not have external nets the alignment during Step (a) is trivially obtained by simply placing all nets with $k_i = 0$ on the leftmost channel positions. The explicit alignment at Step(a) is important for Algorithm 2, given below, which handles external nets and uses Algorithm 1 as a subroutine.

Algorithm 2 is an improvement of the $O(m \cdot \log m)$ time, near optimal algorithm in [4]. W.l.o.g. we assume that the channel instance does not contain external nets with two exits since the terminals of each such net, if any, can be placed anywhere without affecting the channel density. (We consider them as zero terminals.) Let d_l be the channel density when we only consider the external nets with left exits, d_r be the channel density when we only consider the external nets with right exits. Clearly $max\{d_l, d_r\}$ is a lower bound for the channel density. Since we have exits, the above quantity cannot be zero. Assume w.l.o.g. that $d_l \geq d_r$.

Algorithm 2 uses procedure *preprocessing_step* to generate stacks S_1, S_2 and S_3. Then it calls procedure *concatenate*(S_a, S_b, S_c, S_d) which has input parameters stacks S_a, S_b, S_c and output parameter stack S_d. The input parameters of *concatenate* can be any of the stacks S_i, $i \in [1, 3]$. Procedure *concatenate* generates a new stack S_d which contains, from top to bottom, the items of stack S_a (in the same relative order) then the items of stack S_b (in the same relative order), and finally the items of stack S_c (in the same order).

We are now ready to present Algorithm 2. The algorithm considers various cases. In the description below, if a case is handled similarly to a previously described case we omit its description. In [22] we give a more detailed version of Algorithm 2 that describes in detail all the cases. We use $\{\}$ to denote an empty stack. Let a be an integer. We use $|a|$ to denote its absolute value.

Algorithm 2

Input: *A channel with one cell per side and external nets.* **Output:** *A placement of the terminals so that optimal density (as defined in [2, 12, 14, 21]) is obtained.*

```
(a) Sort, using bucketsort, the external nets with left exits according to their values.
    Push them sorted in three stacks $S1_l$, $S2_l$ and $S3_l$ depending on whether their values are
    positive, zero or negative, respectively, so that the top of each stack has net $N_j$ with the
    smallest absolute value of $k_j$ (among the remaining nets in its stack);
    Similarly, create stacks $S1_r$, $S2_r$ and $S3_r$ for the external nets with right exits;

(b) Let $S_l$ be the stack $Si_l$, $i \in \{1, 3\}$, with the minimum sum of the absolute values of the
    nets in the stack; Let $S'_l$ be the other stack; Let $P'_l$ be the channel side where the nets in
    $S'_l$ have more terminals, $P_l$ be the other side;

    (b1) Place all the nets $N_j \in S2_l$, if any, at the leftmost channel positions;

    (b2) if $S_l \neq \{\}$ then pop and place at the leftmost available positions all the nets
         $N_j \in S_l$;
```

```
        (b3) repeat Pop and place nets $N_j \in S'_l$ until side $P_l$ has no more terminals than side
             $P'_l$; (Let $k_i$ denote the value of net $N_i$. Observe that $|\sum_{N_i \in S_l} k_i|$ may be equal to
             $|\sum_{N_i \in S'_l} k_i|$.)
    (c) Repeat (b) for the external nets with right exits; (We now have stacks $S_r$ and $S'_r$, we place
        on the rightmost positions instead of the rightmost positions.)
    (d) if $d_l = 1$ and $d_r = 0$ then
        (d1) if $\exists$ enough zero terminals to be placed opposite to the $k_i$ leftmost positions of the
             only one external net $N_i$ with value $k_i$ then {
                Place them at the leftmost positions; call Algorithm 1; exit; }
        (d2) else {
                call $preprocessing\_step(S_1, S_2, S_3)$; call $concatenate(S1_l, S1, S1_r, SN1)$;
                    call $concatenate(S3_l, S3, S3_r, SN3)$; call $Internal\_Nets(SN1, S2, SN3)$; exit
                    }
    (e) if $d_l = d_r = 1$ or $(d_l = 0, d_r = 1)$ then work as in (d); (For details see [22].)
    (f) (Now $d_l \geq 2$.) Assume that $d_l > d_r$. (The other cases are handled similarly in [22].)
        (f1) if after (a) we have that $S2_l \neq \{\}$) then {
                call $preprocessing\_step(S_1, S_2, S_3)$; call $concatenate(S1_l, S1, S1_r, SN1)$;
                    call $concatenate(S3_l, S3, S3_r, SN3)$; call $Internal\_Nets(SN1, S2, SN3)$; exit
                    }
        (f2) if after (a) $S2_l = \{\}$) then {
            (f2.1) if $S_l = \{\}$ at (b2) then }
                (f2.1.1) Pop net $N_i$ of value $k_i$ from $S'_l$; Place it at the leftmost available
                         positions; (Now $S'_l \neq \{\}$ since we have: (1) $S2_l$ empty, $S_l$ empty, $S'_l$ empty,
                         and (2) $d_l \geq 2$.)
                Let $K_l$ denote the number of terminals placed close to the left exit and on the
                    top channel side minus the number of terminals place close to the left exit
                    and on the bottom side;
                    Assume w.l.o.g. that $K_l$ is positive. (We handle the other case similarly.)
                (f2.1.2) if $\exists$ $K_l$ zero terminals on the bottom side then place them at the
                         leftmost available positions;
                call $preprocessing\_step(S_1, S_2, S_3)$; call $concatenate(S1_l, S1, S1_r, SN1)$; call
                    $concatenate(S3_l, S3, S3_r, SN3)$; call $Internal\_Nets(SN1, S2, SN3)$; exit
                    }
            (f2.2) if $S_l \neq \{\}$ at (b2) then {
                call $preprocessing\_step(S_1, S_2, S_3)$; call $concatenate(S1_l, S1, S1_r, SN1)$; call
                    $concatenate(S3_l, S3, S3_r, SN3)$; call $Internal\_Nets(SN1, S2, SN3)$; exit
                    } }
end of Algorithm 2
```

Theorem 2.1 *Algorithm 2 interchanges the terminals so that a minimum density (as defined in [2, 12, 14, 21]) is obtained in $O(m)$ time.*

The proof of the theorem appears in [22] and is omitted due to space limitations.

2.2 Fixed Terminals on One Side

We now examine the special case proposed by Cong and Khoo [4] where the channel consists of two cells, one cell per channel side, the terminals on one cell (w.l.o.g. the bottom cell) are fixed, and the terminals on the other cell are interchangeable. We show that the problem of obtaining minimum channel density by interchanging the terminals in the top channel cell is NP-hard. We prove the latter by reducing the bin packing problem, defined below, to the decision version of our problem in which we are examining whether there exists a terminal permutation such that the channel density is less than or equal to an input integer K. We call this problem the density decision (*DD*) problem.

Bin packing instance: A finite set U of items, a size $s(u) \in Z^+$ for each $u \in U$, a positive integer bin capacity B, and a positive integer M. *Question:* Is there a partition of U into disjoint sets U_1, U_2, ..., U_M such that sum of the sizes of the items in each U_i, $1 \leq i \leq M$, is B or less?

Theorem 2.2 *The DD problem is NP-complete.*

Proof: Consider an instance of the bin packing problem as described above. Let $|U|$, $|U_i|$, and N_1 denote the size (number of items) of the set U, the size of each set U_i, $1 \leq i \leq M$, and the number of items $u \in U$ with $s(u) = 1$, respectively. We construct the channel instance for the *DD* problem as follows: The length, n, of the constructed channel is equal to $M \cdot B + 2 \cdot (M - 1)$. The top side of the channel has N_1 zero terminals one for each item $u \in U$ with $s(u) = 1$. We call each such terminal an *item zero terminal.* We have $|U| - N_1$ nets, each having terminals in cell C. Each one of these nets corresponds to an item $u \in U$ with size $s(u) \geq 2$. We call such a net an *item net.* For each item u with $s(u) \geq 2$ there exists an item net connecting $s(u)$ terminals. The remaining terminals on the top side are zero terminals. The bottom channel side consists of two terminal nets at the positions described in the following two-nested for-loop.

```
for $k = 1$ to $M - 1$ do {
for $j = 1$ to $B$ do
$\exists$ net with terminals at positions $B_{k\cdot(B+j)}$ and $B_{k\cdot(B+j+1)}$ }
```

The remaining terminals on the bottom side are zero terminals. We call these zero terminals *hole zero terminals* since they do not contribute to the channel density. Any B consecutive hole zero terminals form a *hole* c_i, $1 \leq i \leq M$, which corresponds to set U_i of the bin packing instance. Clearly, from the description of the top channel side we conclude that the nets in the channel connect terminals either on the top or the bottom side but not on both sides. The number K in the *DD* instance is equal to 1. In [22] we have shown that the bin packing problem is satisfied iff the constructed *DD* instance is satisfied. □

3 Other Objective Functions

3.1 Via Minimization

We present an algorithm, we call it *via_minimization*, that minimizes the number of via holes needed for the routing of a channel. Observe that any optimal solution for this problem requires that each net consists of one trunk section [11, 16]. We also assume that we do not have any external nets with two exits and no terminals in the channel since none of these nets requires any vias. Observe also that an alignment of any two terminals of a multiterminal (internal or external) net or an external two terminal net reduces the number of vias by one, but an alignment of the two terminals of a two terminal net reduces the number of vias by two. Thus the problem of minimizing the number of vias is equal to the problem of maximizing the quantity $2 \cdot A_{i2} + A_r$, where A_{i2} is the number of aligned two terminal internal nets and A_r is the number of aligned terminal pairs on any other net. This is a weighted version of the maximum alignment problem that Lin and Sahni examined in [15] for internal nets only. Now the alignment of the two terminals of an internal two terminal net equals to the alignment of two pair of terminals on any other net(s).

Lin and Sahni's algorithm [15], we call it *Lin_Sahni*(C), does not obtain optimal solution to our problem even for a channel C with internal nets only. (In [22] we give a counterexample.) However, Lin and Sahni's algorithm [15] can be serve as a subroutine for a simple algorithm that obtains via minimization. First, we briefly give a straightforward modification of algorithm *Lin_Sahni*(C), we call it *generalized_Lin_Sahni*(C), to obtain maximum alignment of a channel instance C with internal and external nets. (We assume that each external net has at least one terminal on some side of the channel; otherwise it requires no via and it is omitted.)

Procedure *generalized_Lin_Sahni*(C) consists of the following simple channel modification:

```
Set $j = n + 1$;
**while** $\exists$ an external net $N_i$ **do** {
place a new terminal $t_i$ at position $T_j$ and in a new cell; $j = j + 1$ }.
Let $C_1$ be the new channel; **call** *Lin_Sahni*($C_1$)
```

Algorithm 3 is as follows: First obtain maximum alignment in the reduced channel consisting of the two terminal internal nets only. Then contract the columns that have aligned nets. Let C_2 be the reduced channel instance. Call *generalized_Lin_Sahni*(C_2) to obtain via minimization.

Theorem 3.1 *Algorithm 3 obtains via minimization in $O(m)$ time.*

3.2 Maximum Planar Subset of Nets

Our algorithm (Algorithm 4) for the maximum planar subset of nets consists of three phases. The first phase (*phase_1*) transforms the input channel instance to one consisting of internal nets only as follows. The channel has two more cells c_l and c_r at the leftmost and rightmost positions on the (w.l.o.g.) top side. Each external net with a left exit has an additional terminal on cell c_l and each external net with a right exit has an additional terminal on c_r. For simplicity assume that the number of terminals in the new channel is m.

The second phase of the algorithm interchanges the terminals on their cells so that we avoid the intersection between any two nets N_i and N_j if this is possible. This can be done in $O(m)$ time by the following procedure.

```
**procedure** *phase_2*
**Input:** *A channel with interchangeable terminals.* **Output:** *A terminal interchange so that we avoid the intersection between any two nets.*
(a) Transform the channel to form a switchbox $C$ that has no terminals on the vertical sides.
(b) Label the cells $c_1$ to $c_t$, in a clockwise manner by arbitrarily selecting a cell as the origin;
Label in a clockwise manner the terminal positions;
```

(c) Ignore all nets that connect terminals in the same cell; Assume that each net has at most one terminal per cell; (Otherwise all the terminals of the net will be assigned consecutive positions.) For simplicity, assume that we do not have two nets with terminals on exactly the same cells. (If this is the case and the nets are two terminal nets we assign the terminals on consecutive positions in their cells and in a relative order that guarantees that the two nets do not intersect with each other. Otherwise, we assign them on consecutive positions arbitrarily.)
(d) for each cell c_i sort the terminals $t_j \in c_i$ incident to nets N_k, using bucketsorting, according to the net they belong as follows {
(d1) if N_k has already been examined (when we were on cell $c_j, wherej < i$) **then** t_j is in some entry of an array $A(c_i)$. Assign, starting from the larger array entry, all the terminals in the $A(c_i)$ on increasing terminal positions in c_i, starting from the smaller terminal positions.
(d2) All the remaining terminals t_j are sorted according to the cell number c_n of the next terminal t_j^n on N_k. The smaller the cell number the higher the terminal position of t_j in cell c_i. In case of a tie, the relative order depends on whether t_j^n is the last terminal on its net;
if t_j^n is the last terminal of N_k (at most one such terminal can exist after the assumption in step (c)) **then** place t_j on a higher terminal position than all the other terminals in the tie. The positions of the remaining terminals t_j is arbitrary.
(d3) for each net N_k in Step (d2) consider all the terminals t'_j in cells c_j, where $j > i$;
Place each $t'_j \in A(c_j)$ so that if $\exists$ more than one such terminal, each in a different net, the terminal that is the last one on its net is placed on higher entrance in $A(c_j)$; The positions of the remaining terminals will be arbitrary in $A(c_j)$. }

end of *phase_2*

We describe now a dynamic programming approach (last phase of the algorithm) that obtains the minimum planar set of nets in the switchbox C. This is an extension of Supowit's algorithm [18] to multiterminal nets. From C construct a circle graph $G = (V, E)$ so that there exists a vertex for each net and an edge $e \in E$ between vertices v and w iff net v intersects with net w. We also assume a numbering of the terminals in C from 0 to $m-1$. Observe that our problem is equivalent to finding a maximum independent set $I(0, m-1)$ on graph G.

In order to find the maximum independent set on graph G we must first find the maximum independent set $I(i,j)$ for each circle graph $G_{i,j} = (V_{i,j}, E_{i,j})$ which is induced from graph G as follows: $V_{i,j}$ consists of all nets with all terminals having labels in the range $[i,j]$. Edge $e = (v,w) \in E_{i,j}$ iff $v, w \in V_{i,j}$ and net v intersects with net w. We maintain a two dimensional array I. Entrance $I(i,j)$ will have the maximum independent set of graph $G_{i,j}$. $I(i,j_1)$ is computed before $I(i,j_2)$ if $j_1 < j_2$. Observe that in order to find the maximum independent set for graph G we must find $I(0,m-1)$ since $G_{0,m-1} = G$. This is obtained by the following two-nested for loop which is *phase_3* of the algorithm that finds a maximum planar set of nets.

```
for j=0 to M-1 do
for i=0 to M-1 do
if all the terminals $\{k_1, .., k_t, j\}$ of net $N_j$ are in the range $[i,j]$ and $|I(i,k_1-1)| + |I(k_1+1,k_2-1)|$
+ $|I(k_2+1,k_3-1)|$ + ... + $|I(k_t+1,j)| + 1 > |I(i,j-1)|$
then $I(i,j) = I(i,k_1-1) \cup I(k_1+1,k_2-1) \cup I(k_2+1,k_3-1) \cup ... \cup I(k_t+1,j) \cup v_j$
else $I(i,j) = I(i,j-1)$
```

In order to find the nets that participate in entrance $I(0, m-1)$ we must modify the above for loop to decrease the time complexity. Let m be the number of terminals, and D be the maximum number of terminals on a net. We have the following theorem. (For the proof see [22].)

Theorem 3.2 *Algorithm 4 finds a maximum planar subset of nets of an arbitrary chan-*

nel instance with interchangeable terminals in $O(m^2 \cdot D)$ *time.*

References

[1] M.J. Atallah and S. E. Hambrusch, "Optimal rotation problems in channel routing", IEEE Transactions on Computers, vol. C-35, pp 223-232.

[2] Cai, Y. and D. F. Wong, "An optimal Channel Pin Assignment Algorithm", IEEE Transactions on Computer-Aided Design, vol. 10, November 1991, pp.1413-1424.

[3] Cong, J., "Pin Assignment with Global Routing for General Cell Designs", IEEE Transactions on Computer-Aided Design, November 1991, vol. 10, pp. 1401-1412.

[4] Cong, J. and K. Khoo, "A Provable Near-Optimal Algorithm for the Channel Pin-Assignment Problem" Proceedings of International Conference on Computer Design, 1991, pp. 319-321.

[5] Deogun, J. and B. Bhattacharya, "Via Minimization in VLSI Routing with Movable Terminals", IEEE Transactions on Computer-Aided Design, August 1989, vol. 8, pp. 917-920.

[6] Garey, M. R. and D. S. Johnson, "Computers and Intractability: A guide to the theory of NP-completeness", W. H. Freeman and Company, 1979.

[7] Gopal, I. S., D. Coppersmith, and C. K. Wong, "Optimal Wiring of Movable Terminals", IEEE Transactions on Computers, vol. 32, 1983, pp. 845-850.

[8] Hou, C. Y. and C. Y. R. Chen, "A Hierarchical Methodology to Improve Channel Routing by Pin Permutation", Proceedings of International Conference on Computer-Aided Design, 1991, pp. 440-443.

[9] Kobayashi, H., and C. E. Drozd, "Efficient Algorithms for Routing Interchangeable Terminals" IEEE Transactions on Computer-Aided Design, July 1985, vol.4, pp.204-207.

[10] A.S. LaPaugh and R.Y. Pinter, "On minimizing channel density by lateral shifting", In Proceedings 1983 International Conference on Computer-Aided Design, 1983, pp. 121-122.

[11] Lengauer T., "Combinatorial algorithms for Integrated Circuit layout", John Wiley and Sons, 1990.

[12] Leong, H. W. "Routing problems in the physical design of integrated circuits", Ph.D. Thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

[13] Liao K.F. and M. Sarrafzadeh, "Boundary Single Layer Routing with Movable Terminals", IEEE Transactions on Computer-Aided Design, vol. 10, no. 11, 1991.

[14] Leong, H. W. and C. L. Liu, "Permutation Channel Routing", Proceedings of International Conference on Computer Design, 1985, pp. 98-101.

[15] Lin, L. and S. Sahni, "Maximum Alignment of Interchangeable Terminals", IEEE Transactions on Computers, October 1988, vol. 37, pp. 1166-1177.

[16] Preas, B., and M. Lorenzetti, "Physical Design Automation of VLSI Systems", Benjamin/Cummings, 1988.

[17] Pedram, M., M. Marek-Sadowska and E. S. Kuh, "Floorplanning with Pin Assignment", Proceedings of International Conference on Computer-Aided Design, 1990, pp. 98-101.

[18] K. J. Supowit, "Finding a maximum planar subset of nets in a channel", IEEE Transactions on Computer Aided Design, pp. 93-94, vol. 6, no. 1, January 1987

[19] S. Sahni and S. Wu, "Two NP-Hard Interchangeable Terminal Problems", IEEE Transactions on Computer-Aided Design, April 1988, vol. 7, pp. 467-472.

[20] Schlag, M.D.F, L.S. Woo, and C.K. Wong, "Maximizing pin alignments by pin permutations", Integration, the VLSI journal vol.2, no. 4, 1984, pp. 279-307.

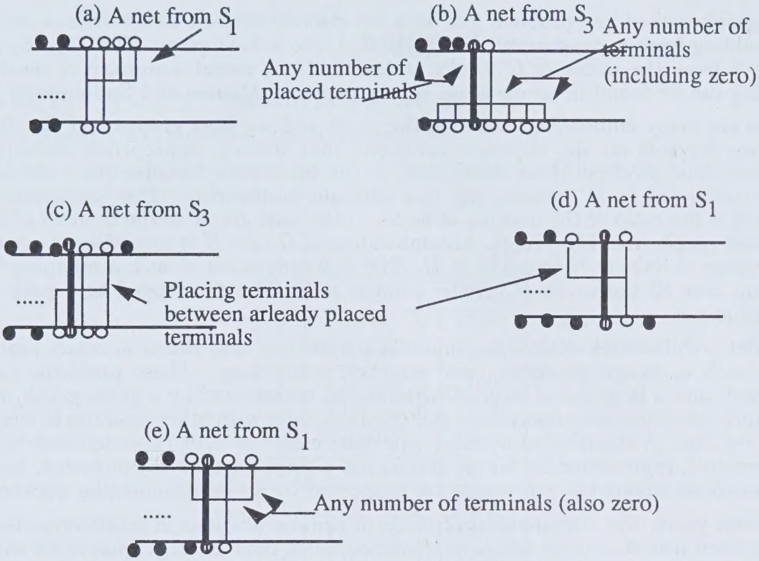
[21] Tollis, I. G., and S. Tragoudas, "River routing and density minimization for channel routing with interchangeable terminals", submitted for journal publication. Part of the paper appears in the Proceedings of the IEEE International Conference on Circuits and Systems, May 1990, pp. 340-347 entitled "Interchanging Terminals for Improved Channel Routing".

[22] S. Tragoudas, "Algorithms for Channel Routing with Interchangeable Terminals", Technical Report 92-10, Department of Computer Science, Southern Illinois University at Carbondale, Carbondale IL 62901.

[23] I.G. Tollis, "On Internal-External layouts", IEEE Transactions on Circuits and Systems, vol. 36, no. 2.

[24] Widmayer P., and C. K. Wong, "An Optimal Algorithm for the Maximum Alignment of Terminals", Information Processing Letters, February 1985, vol. 20, pp. 75-82.

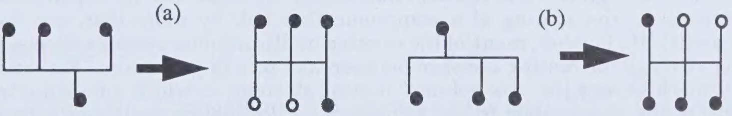
Figure 1
Placed terminals are represented by solid cycles, the terminals of the newly placed net are represented by empty cycles. Shaded circles connected on thick nets represent placed terminals in S_2. (Shaded terminals appear only if the channel has external nets.



(a) A net from S_1
.....
(b) A net from S_3
Any number of placed terminals
Any number of terminals (including zero)
(c) A net from S_3
.....
Placing terminals between arleady placed terminals
(d) A net from S_1
.....
(e) A net from S_1
.....
Any number of terminals (also zero)

Figure 2

A net with (a)positive, (b) negative value and its transformation. Added zero terminals are represented by empty cycles. The terminals are assigned on consecutive positions after the transformation.



EMBEDDING COMPLETE BINARY TREES ON SQUARE MESHES WITH SMALL LINK CONGESTION[1]

Sang-Kyu Lee and Hyeong-Ah Choi
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, DC 20052

The graph embedding problem has been extensively studied under various settings. An embedding from a *guest* graph $G = (V_G, E_G)$ into a *host* graph $H = (V_H, E_H)$ is a mapping f from the nodes of G to the nodes of H. A recent summary of results on embedding can be found in an excellent survey paper by Monien and Sudborough [2].

There are many embeddings between the guest and the host graphs and the desired embedding depends on the objective functions that provide appropriate measures of efficiencies. The *dilation* of an embedding is the maximum distance (i.e., the length of a shortest path) in H between any two adjacent nodes in G. The *expansion* of an embedding is the ratio of the number of nodes in the host graph to the number of nodes in the guest graph (i.e., $|V_H|/|V_G|$). An embedding of G into H is sometimes augmented by a mapping of links in G to paths in H. The *link congestion* of such a mapping is the maximum, over all links e in H, of the number of links in G mapped to a path in H which includes e.

Parallel architectures exploit the inherent parallelism that exists in many problems in areas such as image processing and scientific computing. These problems can be decomposed into a large set of interacting modules, represented by a guest graph, where each module performs some operations and communicates with other modules to compute its final results. A distributed-memory multiprocessors system (also referred to as *a multicomputer*), represented by a host graph, has a fixed number of processors, and the communications among the processors are supported by an interconnection network.

In recent years, the *circuit-switched* mode of communications in multicomputer systems has been introduced for effective transmission of data in large quantities without incurring the communication overhead associated with the packet-switched mode. In the circuit-switched communication, when a source node needs to send a message to its destination node, a dedicated circuit (i.e., a path) is set up between the two nodes, and the message then travels through this path without involving intervening processors. The circuit-switched communication mode has been implemented in various multicomputer systems. These include two types of interconnection networks: the hypercube which is commercially available as iPSC-860 and iPSC-2 machines, and the mesh which is commercially available as Symult 2010 machine.

It was reported that the impact of the path length and the node contention (i.e., the sharing of a node by more than one source-destination paths) in the circuit-switched communication is negligible while the communication overhead heavily depends on the link contention (i.e., the sharing of a communication link by more than one source-destination paths) [1]. Further, many of the existing multicomputer systems allow a single fixed routing strategy for routing message between any pair of processors. For example, Symult 2010 machine uses the 'row-column' routing strategy, in which a message travels along the row of the source node to the column of its destination node and then along the column to the destination node.

Clearly, it is advantageous to minimize the dilation in an embedding when the host graph represents a multicomputer with the packet-switched mode. However, when considering an embedding of a guest graph into a host graph representing a multicomputer with circuit-switched mode, the link congestion becomes a more important factor.

Motivated by the above discussion, this paper considers the problem of embedding a complete binary tree into a square mesh with the objective of minimizing the expansion while the link congestion does not exceed the given value. In particular, we develop a

---

[1] This work was supported in part by the NSF grant CCR91-09512.

unified approach to find an embedding with the small expansion of a complete binary tree into a square mesh with link congestion no larger than the given value λ.

Theorem. A complete binary tree of depth p can be embedded into a $n_\lambda(p)$ x $n_\lambda(p)$ mesh with link congestion no larger than λ, for λ being any given positive integer, such that

$$N_\lambda(p) = n_\lambda(p) \cdot n_\lambda(p) \leq \begin{cases} \frac{\{n_\lambda(2\lambda+2)+1\}^2}{2^{2\lambda+2}} \cdot 2^p & \text{when } p \text{ is even} \\ \frac{\{n_\lambda(2\lambda+3)+1\}^2}{2^{2\lambda+3}} \cdot 2^p & \text{when } p \text{ is odd,} \end{cases}$$

where $n_\lambda(2\lambda+2) = 4 \cdot 2^{\lambda-1}$ and $n_\lambda(2\lambda+3) = 6 \cdot 2^{\lambda-1}$.

Proof : A proof will appear in the full paper. □

Table 1 summarizes the upper bound of the expansion of the embedding obtained in the above theorem for each $\lambda = 1, 2, 3, \cdots, 15$.

λ	$p =$ even	$p =$ odd
1	1.5625	1.5313
2	1.26563	1.22071
3	1.12891	1.17237
4	1.06348	1.14956
5	1.03150	1.13675
6	1.01569	1.13087
7	1.00783	1.12794
8	1.00392	1.12647
9	1.00196	1.12574
10	1.00098	1.12537
11	1.00049	1.12519
12	1.00024	1.12510
13	1.00012	1.12505
14	1.00007	1.12503
15	1.00004	1.12502

Table 1. The upper bound of $N_\lambda(p)/2^p$.

References

[1] S. H. Bokhari, "Communication Overheads on the Intel iPSC-2 Hypercube," ICASE Interim Report 10, May 1990.

[2] B. Monien and H. Sudborough, "Embedding one Interconnection Network in Another," preprint, 1992.

CAPACITY IN A TWO CELL SPREAD SPECTRUM NETWORK

S.V. HANLY

Statistical Laboratory, University of Cambridge
16 Mill Lane, Cambridge, CB2 1SB, UK.
Tel: +44 0223 337944. Fax: +44 0223 337956.

Abstract

This paper reviews recent work on spread spectrum network capacity. The network is shown to have a capacity that is equitably shared amongst the users via power control. This solution is much simpler than the bandwidth partitioning of the FDMA/reuse approach. The capacity gains from using spread spectrum, arising from the averaging and multiplexing, are also emphasised.

MOBILE RADIO; SPREAD SPECTRUM; POWER CONTROL; NETWORK CAPACITY

1 Introduction

In this paper we review some recent work on the capacity of spread spectrum radio networks. The demands a user places on a mobile radio network are inherently variable: interference fluctuations, bursty traffic, and possibly different data rate requirements. We examine the benefits obtained from multiplexing all users onto a spread spectrum channel.

On the uplink of a two cell network there is a nonlinear relationship between the capacities of each cell; this tradeoff between cell capacities happens automatically, demonstrating that spread spectrum with power control is an effective mechanism for sharing network capacity. The received powers on the downlink are inherently more variable as the receivers are mobile. Nevertheless, we prove a law of large numbers showing that in a large scale system the uplink and downlink have the same capacity. With macro-diversity there is a pooling of receiver capacities and adjacent cell interference effectively disappears. A single capacity can be associated with the whole network.

2 The uplink of a two cell network

2.1 The capacity region

The theory of spread spectrum is reviewed in [8] and was first suggested for mobile radio in [1]. New interest in spread spectrum for mobile has been stimulated since the appearance of [3].

Consider the two cell network depicted in figure 1. We have M_0 users in cell 0 and M_1 users in cell 1 and we wish to characterise the capacity region $\mathcal{R}$ given a total bandwidth of W Hz. It is well known that the performance criterion for such a spread spectrum system can be expressed in terms of a signal to noise density ratio. Most commonly the bit error to noise spectral density ratio ($\frac{E_b}{N_0}$) is used ([9]) but here we multiply it by the bit rate and work with the signal power to noise density ratio (SNR). We define $\mathcal{R}$ to be the set of all pairs (M_0, M_1) such that each user can achieve an SNR of α.

That power control in spread spectrum has the capability to equalise the SNRs of the users was first recognised in [7]. We take this approach here.

Suppose that the M_0 users in cell 0 power control so that they are received at cell site 0 at power Q_0 and the M_1 users in cell 1 power control so that they are received at cell site 1 at power Q_1. Note Q_0 and Q_1 will be allowed to vary with M_0 and M_1. The cell 0 users create

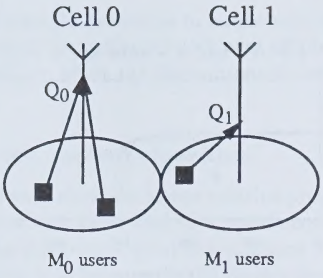


Figure 1: A two cell network

interference at receiver 1 and the cell 1 users create interference at receiver 0. We shall first use the law of large numbers to approximate the cell 0 interference at receiver 1 by $M_0\tilde{\gamma}Q_0$ and the cell 1 interference at receiver 0 by $M_1\tilde{\gamma}Q_1$; $\tilde{\gamma}$ here represents an average fading effect and we assume it is less than unity. The use of the law of large numbers, valid in a large scale system, is a feature of the analysis of spread spectrum schemes. In practice, of course, capacity is reduced by an $O(\sqrt{W})$ term (as $W \uparrow \infty$) because a guarantee must be given on performance quality. For example, if we specify that the powers required to achieve α only exceed the maximum available power a proportion 10^{-6} of the time then we lose some capacity. The law of large numbers is just saying that this is negligible on the $O(W)$ scale. However, it cannot be ignored in practice and we take a large deviations approach in section 2.4. For now we write down the signal to noise density equations as:

$$SNR_0 = \frac{WQ_0}{(M_0-1)Q_0 + M_1\tilde{\gamma}Q_1 + \eta W} = \alpha \tag{1}$$

$$SNR_1 = \frac{WQ_1}{M_0\tilde{\gamma}Q_0 + (M_1-1)Q_1 + \eta W} = \alpha \tag{2}$$

where $\frac{\eta}{2}$ is the spectral density of the external noise, which we assume is white. It is the problem of power control to find the required received powers Q_0 and Q_1. These are given by the equations

$$\begin{aligned} Q_0 &= \frac{\eta W\alpha(M_1\tilde{\gamma}\alpha + W - (M_1-1)\alpha)}{[W-\alpha(M_0-1)][W-\alpha(M_1-1)] - M_0M_1\tilde{\gamma}^2\alpha^2} \\ Q_1 &= \frac{\eta W\alpha(M_0\tilde{\gamma}\alpha + W - (M_0-1)\alpha)}{[W-\alpha(M_0-1)][W-\alpha(M_1-1)] - M_0M_1\tilde{\gamma}^2\alpha^2} \end{aligned} \tag{3}$$

and we can immediately write down the conditions on M_0 and M_1 that enable a solution for Q_0 and Q_1 to be found:

$$\begin{aligned} W-\alpha(M_0-1) &> 0 \\ W-\alpha(M_1-1) &> 0 \end{aligned}$$

and

$$[W-\alpha(M_0-1)][W-\alpha(M_1-1)] - M_0M_1\tilde{\gamma}^2\alpha^2 > 0 \tag{4}$$

We shall denote the set of feasible (M_0, M_1) by $\mathcal{R}$. In section 2.2 we consider the question of power control and exhibit a simple, decentralised algorithm that solves (3), but for now we concentrate on finding a more explicit representation for (4).

For convenience we allow (M_0, M_1) to take nonintegral values and solve for equality in (4) to obtain a boundary to the feasible region. Indeed, even integral points on the boundary are not achievable as infinite received power is required.

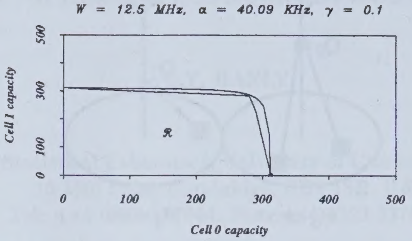


Figure 2: Comparison of capacity regions

It is not difficult to show that the boundary of $\mathcal{R}$ can be parametrised as follows:

$$\mathcal{R} = \{(M_0, M_1) : M_0 = f\left(\frac{W}{\alpha} + 1\right), M_1 = \frac{1-f}{1-f+f\tilde{\gamma}^2}\left(\frac{W}{\alpha}+1\right) \quad 0 \le f \le 1\}$$

Thus if both cells are to have equal numbers then

$$M_0 = M_1 = \frac{1}{1+\tilde{\gamma}}\left(\frac{W}{\alpha}+1\right)$$

at capacity.

It is of interest to compare $\mathcal{R}$ with the capacity region for a system with constant received powers. In [2] a capacity region for a hexagonal cellular system is given. The two cell analogue of this is

$$\begin{aligned} M_0 + \tilde{\gamma} M_1 &\le \frac{W}{\alpha} + 1 \\ \tilde{\gamma} M_0 + M_1 &\le \frac{W}{\alpha} + 1 \end{aligned} \tag{5}$$

Both the capacity regions (4) and (5) are depicted in figure 2, for $W = 12.5$ MHz, the bandwidth allocated to mobile radio and $\alpha = 40.09$ KHz, corresponding to an $\frac{E_b}{N_0}$ requirement of 7 dB as used in [3]. Our choice of $\tilde{\gamma} = 0.1$ is approximately one-sixth of the inter-cell interference term of 0.56 quoted in [8] for a system with six adjacent cells.

Note that although the region with power adaptation is only marginally larger than that with constant powers, adaptive power control has the potential for allowing effects to propagate through the network. The objective is to have a single network-wide capacity that is shared amongst the users via power control. Note that if $M_0 > M_1$ then by (3) $Q_0 > Q_1$ and vice versa. We conjecture that a decrease in received power levels in a large number of cells may allow significant extra capacity in some other part of the network. A low demand in one part of the network may allow a high demand in another part even if the distance would seem to preclude any direct interaction (powers typically fade as the inverse fourth power of distance).

Another point is that our power control can mitigate interference fluctuations, as in section 2.4. In addition, it leads naturally to considering expanding and contracting cells ([6]). As Q_0 increases due to high demand in cell 0, cell 0 contracts and cell 1 expands to take up some of this demand. This is difficult to analyse, but the extension to macrodiversity is examined in section 4. Significant increases in capacity are observed for this more sophisticated power control.

Another reason for considering our capacity region is that a simple, decentralised power control algorithm exists which can be used as a basis for call acceptance and control: if the received power level in one cell becomes high this indicates that the capacity is being saturated in that cell. We suggest that this might generalise to larger networks so that power control itself can share the capacity resources of the entire network over all the users. In section 2.4 we shall consider how this power control can offset the fluctuations in interference so that performance guarantees can be made.

2.2 A decentralised power control algorithm

In [7], the SNR balancing problem is shown to have a solution provided a function of the required SNR does not exceed the largest real eigenvalue of a matrix corresponding to the power gains of each user. The corresponding eigenvector gives the solutions for the transmitted powers.

The idea behind our algorithm is to balance the SNRs in a decentralised way: let Q_i increase if the users in cell i have an SNR below α and let Q_i decrease if the users have an SNR above α. At any given time there is a received power that will achieve α in the current interference environment. In cell 0 we need a received power level of $\frac{\alpha M_1\tilde{\gamma}Q_1+\eta W\alpha}{W-\alpha(M_0-1)}$ where Q_1 is the current received power level in cell 1, and in cell 1 we need a received power level of $\frac{\alpha M_0\tilde{\gamma}Q_0+\eta W\alpha}{W-\alpha(M_1-1)}$. Consider the following differential equations:

$$\dot{Q} = f(Q)$$

where f is defined by

$$f\begin{pmatrix} Q_0 \\ Q_1 \end{pmatrix} = \rho \begin{pmatrix} \frac{\alpha M_1\tilde{\gamma}Q_1+\eta W\alpha}{W-\alpha(M_0-1)} - Q_0 \\ \frac{\alpha M_0\tilde{\gamma}Q_0+\eta W\alpha}{W-\alpha(M_1-1)} - Q_1 \end{pmatrix} \tag{6}$$

and ρ is a scale factor with units of Hz.

Lemma 1 *The ODE (6) has a unique, globally stable equilibrium point given by (3) provided* $(M_0, M_1) \in \mathcal{R}$

Proof
It is trivial to verify that an equilibrium point must be a solution to (3). To prove global stability we note that the system (6) is linear and ∇f has eigenvalues

$$-1 \pm \frac{\alpha\sqrt{M_0 M_1 \tilde{\gamma}^2}}{\sqrt{[W-\alpha(M_0-1)][W-\alpha(M_1-1)]}}$$

Thus, provided $(M_0, M_1) \in \mathcal{R}$ and hence satisfies (4), both eigenvalues are negative and the system is globally stable. •

The algorithm will seek out the correct received powers (Q_0, Q_1) for a given (M_0, M_1). The speed of convergence in real time depends on the value of the scale parameter ρ, itself determined by hardware. As mobiles arrive, depart and handover, the algorithm allows the powers Q_0 and Q_1 to adapt so as to keep all SNRs at α. Note that the closer the number of users is to the boundary of $\mathcal{R}$ the slower the convergence. Note also that as the boundary is approached the equilibrium received powers tend to infinity. A phase portrait is depicted in figure 3.

The paper [12] has recently been brought to our attention. It takes the same approach as that of [7], namely equalising the SNRs of the users, but it does so in a decentralised way. The algorithm utilises the power method for finding the dominant eigenvalue of a matrix. Neither paper includes external noise and so the solutions for powers are not unique; this creates problems such as power escalation that are not encountered in our algorithm. Neither do they relate network capacity and algorithm stability. On the other hand, both papers deal with general cellular networks, rather than just the two cell case.

We have assumed so far that the fluctuations in interference arising from fading and voice activity can be neglected. In practice there will be power fluctuations on a fast time scale to cope with the fluctuations in interference not included in the model. We consider this issue in section 2.4. The mobile transmitter powers themselves will be highly variable in order to cope

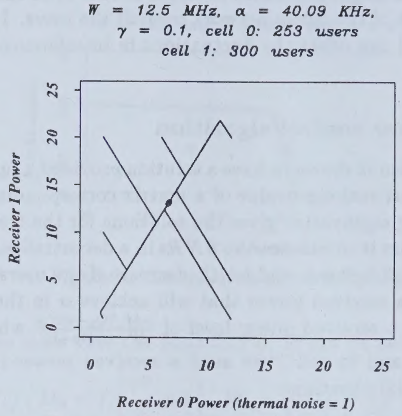


Figure 3: Phase portrait for power control

with the changing environment of the mobile (this of course depends on the level of mobility). In the next section we consider the constraint on capacity imposed by a maximum transmitter power level.

2.3 Transmitter power level constraints

A maximum transmitter power level must reduce the capacity region, since Q_0 and Q_1 tend to infinity as the boundary of $\mathcal{R}$ is approached. Suppose that $M_0 = M_1$ and that a mobile transmitting at full power in the worse case position is received at power Q. Then we can solve (3) for M and find that

$$M = \frac{1}{1+\bar{\gamma}}\left(\frac{W}{\alpha} + 1 - \frac{\eta W}{Q}\right) \tag{7}$$

The reduction in capacity is $\frac{\eta W}{Q}$ and this can be several mobiles in a large bandwidth system. For example, a typical value of $\frac{Q}{\eta W}$ in a 1.25 MHz system is -1 dB, according to [3], and so in a 12.5 MHz system it is about -11 dB. Thus the reduction term $\frac{\eta W}{Q}$ is about 12 in a 12.5 MHz system. Of course, in a large bandwidth system this will be small in comparison to $\frac{W}{\alpha}$. This suggests that we can expect to be close to the boundary of the capacity region and not exceed the maximum power constraint. Indeed, a single user transmitting to a receiver in noise of spectral density $\eta/2$ needs a received power of $\eta\alpha$ to achieve a SNR of α. Let

$$Q = K\left(1 + \frac{\alpha}{W}\right)^{-1} \eta\alpha \approx K\eta\alpha$$

Then

$$\frac{W}{\alpha} + 1 - \frac{\eta W}{Q} = \left(1 - \frac{1}{K}\right)\left(\frac{W}{\alpha} + 1\right)$$

so that only a moderate multiple K of the basic power level $\eta\alpha$ is needed to get very close to capacity.

We stated above that Q corresponds to a worst case mobile. This was taken for the sake of argument and corresponds to no tolerance for degraded performance. In practice, one will guarantee performance a certain proportion of the time, perhaps $1 - 10^{-6}$, for when power level fluctuations and voice activity are considered it becomes impossible to guarantee performance all the time. Certain locations with very strong fading may occur with a low enough frequency to ignore; a large deviations analysis is required.

2.4 Interference fluctuations

In this section we consider an $O\left(W^{\frac{1}{2}}\right)$ reduction in capacity so that the probability that a fluctuation in interference renders the system infeasible is negligible. We use power control to counteract these fluctuations in interference.

Suppose for simplicity that both cells have M users. In each cell the interference from the other cell has mean $M\tilde{\gamma}Q$ but the fluctuations are $O(\sqrt{M})Q$. Consider the worst case in which both fluctuations are above $M\tilde{\gamma}Q$ by the same amount, which we denote by FQ. The signal to noise density ratio in both cells is then

$$\frac{WQ}{(M-1)Q+M\tilde{\gamma}Q+FQ+\eta W}$$

and in the limit as $Q\uparrow\infty$ we get the limiting capacity of

$$\frac{1}{1+\tilde{\gamma}}\left(\frac{W}{\alpha}+1-F\right)$$

users. In fact, it is clear from (7) that with a maximum received power constraint of Q the capacity is

$$M=\frac{1}{1+\tilde{\gamma}}\left[\frac{W}{\alpha}+1-F-\frac{\eta W}{Q}\right]$$

Note that F is $O(\sqrt{W})$ and $\frac{\eta W}{Q}$ is $O(W)$ so as W grows large the fluctuations from other users become less and less significant and the external noise becomes the dominating effect.

Consider now the case that the fluctuations in interference are of different magnitudes. Let the number in each cell be M but suppose that F_0 is the value for the interference from cell 0 and F_1 is the value for the interference from cell 1. Then the signal to noise density equations are

$$\begin{aligned}\frac{WQ_0}{(M-1)Q_0+M\tilde{\gamma}Q_1+F_1Q_1+\eta W}&=\alpha\\ \frac{WQ_1}{(M-1)Q_1+M\tilde{\gamma}Q_0+F_0Q_0+\eta W}&=\alpha\end{aligned}$$

It is shown in [6] that a sufficient condition for a solution in Q_0, Q_1 existing is

$$M=\frac{1}{1+\tilde{\gamma}}\left(\frac{W}{\alpha}+1-\left(\frac{F_0+F_1}{2}\right)\right)$$

The conclusion is that by using power control we only need to ensure that the **average** of interference fluctuations does not go too high. Thus the large deviations approach is the following. Let M be defined by

$$M=\frac{1}{1+\tilde{\gamma}}\left[\frac{W}{\alpha}+1-F-\frac{\eta W}{Q}\right]$$

where Q is the maximum received power level. We choose F so that the average of the fluctuation in each cell goes above F only a tiny proportion of the time, given by the performance guarantee. When the average fluctuation does go above F, Q is not sufficient to ensure a signal to noise density ratio of α and so performance is degraded.

3 The downlink of a two cell system

The downlink requires a more sophisticated power control that can adapt very quickly as the interference environment of a mobile changes. Of course, the transmitted power level on the uplink has a high variance as well, but the received power levels vary only slowly. On the

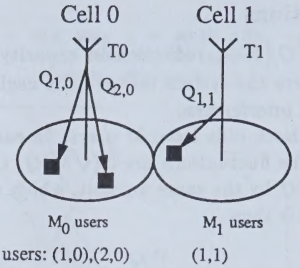


Figure 4: A two cell network

downlink the danger is that the fluctuations in interference will render the system infeasible, but we show that the probability of infeasibility can be made very small in a large scale system.

Consider the two cell system depicted in figure 4.

Label the users in cell 0 $(1,0)\cdots(M_0,0)$ and those in cell 1 $(1,1)\cdots(M_1,1)$. Let $Q_{i,0}$ be the transmitted power of the signal destined for user $(i,0)$ by cell site 0. Similarly, let $Q_{i,1}$ be the transmitted power of the signal destined for user $(i,1)$ by cell site 1. Let $\gamma_{0,(i,0)}Q_{i,0}$ be the received power at user i of the signal intended for that user and let $\gamma_{1,(i,0)}Q$ be the interference at user $(i,0)$ of a signal from cell site 1 transmitted at power Q. Then the signal to noise density ratio at user $(i,0)$ is given by

$$SNR_{(i,0)} = \frac{W\gamma_{0,(i,0)}Q_{i,0}}{\sum_{j=1}^{M_0} I[j \neq i]Q_{j,0}\gamma_{0,(i,0)} + \sum_{j=1}^{M_1} Q_{j,1}\gamma_{1,(i,0)} + \eta W}$$

Assume

$$\begin{aligned} \mathsf{E}\left[\frac{\gamma_{0,(i,1)}}{\gamma_{1,(i,1)}}\right] &= \mathsf{E}\left[\frac{\gamma_{1,(i,0)}}{\gamma_{0,(i,0)}}\right] \equiv \tilde{\gamma} \\ \mathsf{E}\left[\gamma_{0,(i,0)}^{-1}\right] &= \mathsf{E}\left[\gamma_{1,(i,1)}^{-1}\right] \equiv \tilde{\mu} \end{aligned}$$

In [6] it is shown that it is required that

$$[W-(M_0-1)\alpha]\,[W-(M_1-1)\alpha] - \sum_{i=1}^{M_0}\sum_{j=1}^{M_1} \frac{\gamma_{0,(j,1)}}{\gamma_{1,(j,1)}}\frac{\gamma_{1,(i,0)}}{\gamma_{0,(i,0)}}\alpha^2 > 0 \tag{8}$$

in order for the SNR equations to have a solution. We wish to place a constraint on the number of users to prevent the left hand side of (8) from becoming negative (or zero). In section 2.1 we have shown that (4) describes the set of feasible (M_0, M_1) on the uplink. Maximum power levels and the consideration of interference fluctuations reduces this set as described in sections 2.3 and 2.4. Here, interference fluctuations are much more significant as the receivers are now mobile. Indeed, if M_0 and M_1 are $O(W)$ then the variance of (8) is $O(W^3)$, so we must constrain M_0 and M_1 in order to bound the left hand side of (8) away from zero. We remark that (8) corresponds to the denominator of the total transmitted power at either receiver, so the transmitted powers tend to infinity as the left hand side of (8) tends to zero. Our approach is to impose a power constraint and obtain a law of large numbers showing that the average power used per signal will tend to a constant. For simplicity, we again assume that $M \equiv M_0 = M_1$; we want to bound M by imposing feasibility and power constraints. Fix Q and let $M(W)$ be defined by

$$M(W) = \frac{1}{1+\tilde{\gamma}}\left[\frac{W}{\alpha} + 1 - \frac{\eta W \tilde{\mu}}{Q}\right]$$

Then as $W \uparrow \infty$ we obtain the following central limit theorem ([6])

Theorem 1 *As $W \uparrow \infty$ the probability of infeasibility (no solution for powers) tends to zero, the average per-user transmitted power tends to Q and the total transmitted power in each cell is*

$$M(W)Q + O(W^{1/2})$$

4 Two receiver macro-diversity

It seems reasonable to consider systems in which distributed coding takes place. This is because it is the frequency spectrum that is the beleaguered resource, not communication between receivers connected by, say, optical fibres. The main motivation is to remove the other-cell interference (including interference fluctuations). Macrodiversity is considered in [3] and [10] where a mobile near a cell boundary can commence cell site diversity mode and be processed by several cell sites simultaneously. All cell sites forward their demodulated data to the system controller which chooses the signal of the highest quality. Our approach is to use all signals in the demodulation so that distributed coding actually takes place.

Consider the uplink of a two receiver network depicted in figure 5.

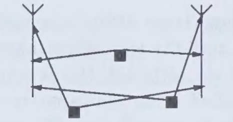


Figure 5: Two receiver macro-diversity

Let there be M users attempting to communicate with receivers 0 and 1. Let Q_i be the transmitted power of user i and $\gamma_i^0 Q_i$, $\gamma_i^1 Q_i$ the received powers at receivers 0 and 1 respectively. Define $Q_0^{(t)}$,$Q_1^{(t)}$ by

$$\begin{aligned} Q_0^{(t)} &\equiv \sum_{i=1}^{M} \gamma_i^0 Q_i \\ Q_1^{(t)} &\equiv \sum_{i=1}^{M} \gamma_i^1 Q_i \end{aligned}$$

so that $Q_0^{(t)}$ and $Q_1^{(t)}$ are the total received powers at each receiver.

What precisely is signal to noise density in a two receiver macro-diversity system? It is shown in Appendix A of [5] that the sum of the signal to noise density ratios is the relevant performance criteria when signal to noise power is low (true for spread spectrum, not for FDMA/reuse). This result is generalised in [4] to arbitrary numbers of receivers. Thus the SNR requirement is that

$$\frac{\gamma_i^0 Q_i W}{Q_0^{(t)} + \eta W} + \frac{\gamma_i^1 Q_i W}{Q_1^{(t)} + \eta W} = \alpha \quad i = 1 \cdots M$$

It is shown in [5] that these M equations can be solved simultaneously for the Q_i provided that M is sufficiently small.

Theorem 2 *The capacity of the two receiver network with a signal to noise density ratio requirement of α is $\frac{2W}{\alpha}$.*

The interpretation is that the total bandwidth is $2W$ and each user requires a bandwidth of α Hz. It is shown in [5] that each receiver has a bandwidth of W and that as capacity is approached these bandwidths are pooled. It is also shown there that a decentralised power control algorithm converges to the powers that satisfy the SNR equations, provided that the system is below capacity.

It is shown in [5] that further constraints are imposed on network capacity by maximum transmitter power levels. Nevertheless, the overall constraint bites provided that traffic is sufficiently uniform. We conclude that a network-wide notion of capacity exists. We suggest that the adaptive cellular power control algorithm of section 2.2 is a first step towards this objective and that it could be enhanced by allowing cells to expand and contract.

5 Conclusions

We conclude that capacity can be equitably shared amongst the users in a two cell network via power control. The power control algorithm exhibited has the virtue of simplicity; indeed, it is decentralised in nature, only requiring a transmitter to know its current signal to noise density ratio (SNR). This should be contrasted with the difficult task of allocating capacity fairly with dynamic channel assignment in FDMA/reuse systems ([11]). Of course, our power control algorithm requires hardware that can detect and react quickly to changes in interference levels. Whether this can be achieved in practice is an engineering question that we have not pursued.

We have taken various approaches (large deviations and the law of large numbers) which indicate that power control can deal with the variability inherent in a mobile radio network. We have not considered voice activity at all, although this is where most variability comes from. A large deviations approach can be applied using the capacity region defined in the present paper to obtain the capacity region of the voice activity model.

Note that spread spectrum takes automatic advantage of voice activity. It is shown in [5] how the cost of the variability in activity decreases with the scale of the system. This is due to averaging and the law of large numbers. The law of large numbers implies that the single cell capacity of C is reduced to $\frac{1}{1+\tilde{\gamma}}C$ with the presence of an adjacent cell, where $\tilde{\gamma}$ is an **average** interference effect. In FDMA/reuse systems the reuse factor reduces capacity and this is based on **worst case** considerations. We have not made direct comparisons with FDMA/reuse as this seems more relevant in the context of a larger network, but it is apparent that spread spectrum has many advantages from a spectrum efficiency point of view ([3], [5], [7]).

In the final example of a two receiver network with macrodiversity the whole network becomes a single channel. In this case each receiver has a capacity of $\frac{W}{\alpha}$ users. Without diversity the per cell capacity was seen to be $\frac{1}{1+\tilde{\gamma}}\frac{W}{\alpha}$ where $\tilde{\gamma}$ gives the average interference from the adjacent cell. With macrodiversity interference is reduced; this is because all signals carry information. This benefit from macrodiversity does not transfer across to FDMA/reuse: signal to noise powers must be small. This is not to say that some small gains cannot be made from macrodiversity in FDMA/reuse. We have hardly touched on the other benefits of multiplexing that occur when traffic fall into separate classes with different rates and in particular when traffic is bursty.

Acknowledgements The author is very grateful to Frank Kelly, Professor of the mathematics of systems at Cambridge university, for many useful discussions. The financial support of the Commonwealth Scholarship and Fellowship Plan is also gratefully acknowledged.

References

[1] Cooper, G. R., Nettleton, R. W. (1978) A spread spectrum technique for high-capacity mobile communications *IEEE Trans. Vehic. Tech.* VT-27, Nov.

[2] Everitt, D., McMillan, D., Lavery, B. (1991) Performance issues for cellular networks with code division multiple access *Proceedings of the sixth Australian teletraffic research seminar*

[3] Gilhousen, K. S., Jacobs, I. M., Padovani, R., Viterbi, A. J., Weaver, L. A., Wheatley, C. E. (1990) On the capacity of a cellular CDMA system *I.E.E.E. Transactions on Vehicular Technology Vol. 40 No. 2 May 1991.*

[4] Hanly S.V. and Whiting P. (1992) Information-Theoretic Capacity of Multi-Receiver Networks *to appear in Telecommunications Systems*

[5] Hanly S.V. (1991) Capacity and power control in a two receiver network *submitted for publication*

[6] Hanly, S.V. (1992) Capacity and power control in a two cell spread spectrum network *In preparation*

[7] Nettleton, R. W., Alavi H. (1983) Power control for spread spectrum cellular mobile radio system *Proc. IEEE Veh. Tech. Conf.*:242–246

[8] Pickholtz R. L., Schilling D. L., Milstein L. B. (1982) Theory of Spread-Spectrum Communications-A Tutorial *IEEE Transactions on Communications* Com-30:5:855–884

[9] Proakis, J.G. (1989) Digital Communications (second edition) *McGraw-Hill*

[10] Salmasi, A, Gilhousen K. S. (1991) On the system design aspects of code division multiple access (CDMA) applied to digital cellular and personal communications networks *41st IEEE Vehicular Technology Conference, St. Louis Missouri, May 1991*

[11] K.N. Sivarajan (1990) Spectrum Efficient Frequency Assignment for Cellular Radio *PhD Thesis, California Institute of Technology*

[12] Zander, Z. (1991) Near-optimum transmitter power control in cellular radio systems *Radio Communications Systems, Royal Institute of Sweden, research report*

Diversity based detectors in Multi-Access Networks *

Srinivas Kandala , Elvino S. Sousa and Subbarayan Pasupathy
University of Toronto, Department of Electrical Engineering
10 King's College Road, Toronto, Ontario M5S 1A4 Canada
email:srini@comm.toronto.edu phone:(416)978-3651

September 28, 1992

Abstract

We consider a multiple-access channel system in an additive white Gaussian noise (AWGN) environment. We highlight a diversity scheme which is already in place in most cellular networks. We propose detection schemes for this type of system and show how one can take advantage of diversity along with the concept of multi-user detection. These types of detectors should provide improved network capacities, especially in fading environments.

1 Introduction

In this paper, we consider the detection of signals in an additive white Gaussian noise channel. In these detection schemes, signaling information from all the sources is used in estimating the data symbols [1] [2] [3] [4] [5]. Even though the complexity of these receivers is higher than that of a matched-filter receiver (also referred to as the conventional receiver), they are cost-effective when all the users' data has to be estimated simultaneously; for example, in the uplink of a cellular mobile radio environment.

In this paper, we suggest methods to improve the performance (in terms of error probabilities) of such a cellular network [6]. Consider the scenario shown in Fig.1 where an user at the border of two cells is attempting to transmit data. Typically, only one of the base stations accepts data from the user. However, there might be other users at close-proximity to this base station which may cause significant interference (the near-far effect). This effect would be more deleterious if the signal suffers from fading. In this paper, we propose some detecting schemes which would improve the performance.

In section 2, we describe the system model and then suggest some detectors in section 3. These improvements are based on diversity that might be achieved without much increase in complexity. In section 4, we briefly describe how to simplify these detectors and in section 5, we study the performance of these detectors.

---

*This research was supported by a grant from the Information Technology Research Centre, a Centre of Excellence, funded by the province of Ontario, Canada

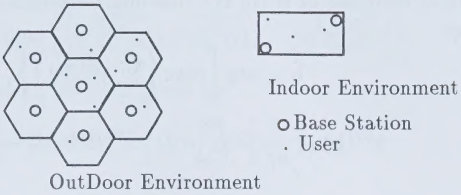


Figure 1: Cellular Systems

2 System Model

The general received signal, $r(t)$, at the base station may be expressed as

$$r(t) = \sum_{i=-M}^{M} \sum_{k=1}^{K} b_k(i)\sqrt{E_k(i)}\, s_k(t - iT - \tau_k) \; + \; n(t). \tag{1}$$

where K is the number of users, T is the symbol interval, $s_k(t)$, E_k, and $b_k(i)$ are the respective modulating signal, energy, and data symbol associated with the k^{th} user during the i^{th} symbol interval, with $s_k(t)$ being zero for $t \notin [0, T]$ and having unit energy. τ_k is the delay associated with the k^{th} user (without loss of generality, we assume $0 \leq \tau_k < T$). For simplicity, we let the data symbols assume values from the alphabet $A = \{-1, +1\}$ (i.e., binary signaling). $n(t)$ is AWGN with two-sided power spectral density of σ^2. In certain cases (for example, in a CDMA system where the difference in delays between the users' signals are less than one chip time), we can assume that all the signals are received at the receiver simultaneously, i.e., $\tau_1 = \cdots = \tau_K = \tau$. We refer to these systems as synchronous systems. If this condition is not satisfied, we refer to such systems as asynchronous systems.

The detectors referred to before use the above signal to estimate the data symbols. Usually, the data would be estimated at the base station and then sent to the switch on a highly reliable channel. Instead of decoding the data at the base station (henceforth referred to as a sensor), the sufficient statistic can be sent to the switch directly where the decoding process can be carried out. This will increase the amount of data to be sent from sensor to the switch, but this increase can be justified by the improvement in the overall performance. We propose to do this by the following procedure: a user is tracked by all the sensors where its power is significant; compute the statistics at every sensor and send these statistics to the switch The switch will combine these statistics and obtains an estimate of the data.

Without loss of generality, we assume that there are K users and that all the users are tracked by the L base stations that are available in the vicinity. Throughout this discussion, we will be considering both asynchronous and synchronous systems.

Asynchronous Systems: The received signal, $r^{(l)}(t)$, at the l^{th} base station can be expressed as,

$$r^{(l)}(t) = \sum_{i=-M}^{M} \sum_{k=1}^{K} b_k(i)\sqrt{E_k^{(l)}(i)}\, s_k(t - iT - \tau_k^{(l)}) \; + \; n^{(l)}(t). \tag{2}$$

Here, we add a superscript l to all the parameters which depend on the sensor. Further, we can also assume that $n^{(l)}(t), l = 1, \cdots, L$ are independent AWGN processes with two-sided power spectral density of σ_l^2. Let $\mathbf{b} = \{\mathbf{b}(i) = [b_1(i), \cdots, b_K(i)]^T; i = -M, \cdots, M\}$, then, the optimal estimate of **b** (in the maximum-likelihood sense) can be found easily and is given by

$$\hat{\mathbf{b}} = \arg\left\{\max_{\mathbf{b}}\left(\sum_{l=1}^{L}\gamma^{(l)}(\mathbf{b})\right)\right\}, \tag{3}$$

$$\text{where} \qquad \gamma^{(l)}(\mathbf{b}) = \frac{1}{\sigma_l^2}\left[2\int_{-\infty}^{\infty} r_l(t)\, S^{(l)}(t;\mathbf{b})\, dt - \int_{-\infty}^{\infty}\left(S^{(l)}(t;\mathbf{b})\right)^2 dt\right], \tag{4}$$

with

$$S^{(l)}(t;\mathbf{b}) = \sum_{i=-M}^{M}\sum_{k=1}^{K} b_k(i)\sqrt{E_k^{(l)}(i)}s_k(t - iT - \tau_k^{(l)}).$$

It can be recognized that $\gamma^{(l)}(\mathbf{b})$ is the required likelihood function for data detection at the l^{th} sensor when this sensor is considered alone for the purpose of detection. Since the required likelihood function is just a linear combination of $\gamma^{(l)}(\mathbf{b})$, one can use Verdú's [1] procedure (by using the Viterbi algorithm) to achieve optimum detection. However, since the complexity of this detector increases exponentially with the number of users, this method is impractical. Hence, in the ensuing discussion, we will consider some sub-optimum detectors.

Synchronous Systems: For synchronous systems, since we assume that $s_k(t)$ is zero for $t \notin [0, T]$, the received signal, for the detection of the i^{th} symbol, can be expressed as

$$r^{(l)}(t) = \sum_{k=1}^{K} b_k(i)\sqrt{E_k^{(l)}(i)}s_k(t - iT - \tau^{(l)}) \; + \; n^{(l)}(t). \tag{5}$$

where we set $\tau_1^{(l)} = \cdots = \tau_K^{(l)} = \tau^{(l)}$. As with the asynchronous systems, the computational complexity of the optimal detector is exponential in number of users, K.

3 Sub-Optimal Detectors

In [1], it was shown that the outputs of a bank of matched filters sampled at the symbol epochs, each filter being matched to one of the users' signals along with the symbol epochs, constitute a sufficient statistic. Using a similar procedure, it can be again shown that the sampled outputs of the bank of matched filters and the symbol epochs do indeed provide a sufficient statistic, even when there is more than one sensor.

Asynchronous Systems: Let $y_k^{(l)}(i)$ denote the output of the matched filter sampled at the symbol epochs (at $iT + \tau_k^{(l)}$) for the i^{th} symbol from the k^{th} user at the l^{th} sensor, i.e.,

$$y_k^{(l)}(i) = \int_{\tau_k^{(l)}+iT}^{\tau_k^{(l)}+iT+T} s_k(t - iT - \tau_k^{(l)})\, r^{(l)}(t)\, dt. \tag{6}$$

Let $\mathbf{y}^{(l)}(i) = [y_1^{(l)}(i) \cdots y_K^{(l)}(i)]^T$, then

$$\begin{aligned}\mathbf{y}^{(l)}(i) \;&=\; \mathbf{R}^{(l)}(-1)\mathbf{E}^{(l)}(i+1)\mathbf{b}(i+1) + \mathbf{R}^{(l)}(0)\mathbf{E}^{(l)}(i)\mathbf{b}(i)\\ &\quad + \mathbf{R}^{(l)}(1)\mathbf{E}^{(l)}(i-1)\mathbf{b}(i-1) + \mathbf{n}^{(l)}(i)\end{aligned} \tag{7}$$

with the $(k,j)^{th}$ element of $\mathbf{R}^{(l)}(i)$ being given by

$$\begin{aligned}R_{kj}^{(l)}(i) \;&=\; \int_{-\infty}^{\infty} s_k(t - \tau_k^{(l)})\, s_j(t + iT - \tau_j^{(l)})\, dt\\ &=\; \begin{cases} 0 & \text{if } \mid i \mid > 1 \\ R_{jk}^{(l)}(-i) & \text{if } i = \pm 1 \end{cases},\end{aligned} \tag{8}$$

and

$$\mathbf{E}^{(l)}(i) = \operatorname{diag}\left(\left[\sqrt{E_1^{(l)}(i)}, \sqrt{E_2^{(l)}(i)}, \cdots, \sqrt{E_K^{(l)}(i)}\right]\right)$$

, The cross-correlation matrix of $\mathbf{n}^{(l)}(i)$ is given by

$$\mathrm{E}\left[\mathbf{n}^{(l)}(i)\,\mathbf{n}^{(m)}(j)^T\right] = \sigma_l^2\,\mathbf{R}^{(l)}(i-j)\,\delta_{lm}$$

, where

$$\delta_{lm} = \begin{cases} \mathbf{I}_K & \text{if } l = m \\ \mathbf{0} & \text{if } l \neq m \end{cases}.$$

As was done in [2], for $l = 1, \cdots, L$ we express (7) as

$$\mathbf{y}^{(l)} = \mathcal{R}^{(l)}\mathcal{E}^{(l)}\mathbf{b} + \mathbf{n}^{(l)}, \tag{9}$$

where

$$\mathbf{y}^{(l)} = \left[\mathbf{y}^{(l)}(-M)\cdots\mathbf{y}^{(l)}(M)\right]^T, \ \mathbf{n}^{(l)} = \left[\mathbf{n}^{(l)}(-M)\cdots\mathbf{n}^{(l)}(M)\right]^T,$$
$$\mathcal{E}^{(l)} = \operatorname{diag}\left(\left[\mathbf{E}^{(l)}(-M), \cdots, \mathbf{E}^{(l)}(M)\right]\right),$$

and

$$\mathcal{R}^{(l)} = \begin{bmatrix} \mathbf{R}^{(l)}(0) & \mathbf{R}^{(l)}(-1) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{R}^{(l)}(1) & \mathbf{R}^{(l)}(0) & \mathbf{R}^{(l)}(-1) & & \vdots \\ \mathbf{0} & \mathbf{R}^{(l)}(1) & \mathbf{R}^{(l)}(0) & \ddots & \mathbf{0} \\ \vdots & & \ddots & \ddots & \mathbf{R}^{(l)}(-1) \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{R}^{(l)}(1) & \mathbf{R}^{(l)}(0) \end{bmatrix}.$$

Synchronous Systems: For the synchronous systems, we may drop the time dependency, since for detecting any symbol, we need the information of symbols transmitted by the other users only during that particular interval. Hence,

$$\mathbf{y}^{(l)} = \mathbf{R}^{(l)}\mathbf{E}^{(l)}\mathbf{b} + \mathbf{n}^{(l)}, \tag{10}$$

where all the vectors are defined in analogous manner, $\mathbf{R}^{(l)} = \mathbf{R}^{(l)}(0)$, and

$$\mathbf{E}\left[\mathbf{n}^{(l)}\mathbf{n}^{(m)T}\right] = \sigma_l^2\mathbf{R}^{(l)}\delta_{lm}. \tag{11}$$

3.1 Conventional Detectors

The most commonly used detector is the matched-filter receiver which we will be referring to as the Conventional Detector (see Fig.2). If there were no diversity, to estimate the i^{th} symbol of the k^{th} users, the decisions would be made based on the rule:
Synchronous Systems:

$$\hat{b}_k^c = \operatorname{sgn}\left\{y_k^{(l)}\right\}, \tag{12}$$

Asynchronous Systems:

$$\hat{b}_k^c(i) = \operatorname{sgn}\left\{y_k^{(l)}(i)\right\}. \tag{13}$$

When diversity is available, one would like to combine all the signals. There are two primary ways of achieving this: Hard decoding and Soft decoding. In this paper, we concentrate on soft decoding. In soft decoding, all the matched-filters' outputs are

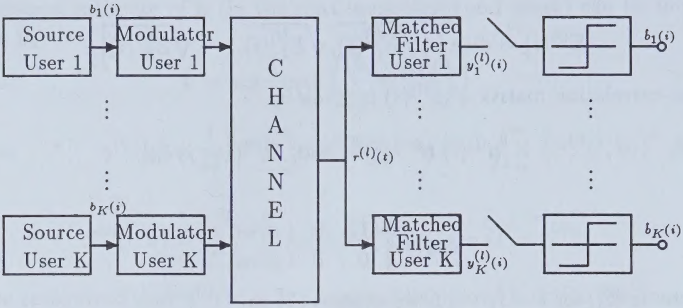


Figure 2: A Conventional Detector

combined and then a decision is made. Further, we consider only the linear combinations of these outputs. Hence, the decisions would be based on the rule:
Synchronous Systems:

$$\hat{b}_k^c = \mathrm{sgn}\left\{\sum_{l=1}^{L} \nu_k^{(l)} y_k^{(l)}\right\}, \tag{14}$$

Asynchronous Systems:

$$\hat{b}_k^c(i) = \mathrm{sgn}\left\{\sum_{l=1}^{L} \nu_k^{(l)}(i) y_k^{(l)}(i)\right\}. \tag{15}$$

One would like to select $\nu_k^{(l)}$ such that the error probability is minimized. However, such a selection is not easy. One way to combine is by using the optimal combining rule as if there were only one user. Then,

$$\nu_k^{(l)}(i) = \frac{\sqrt{E_k^{(l)}(i)}}{\sigma_l^2}. \tag{16}$$

In this paper, for comparison purposes, we always consider the conventional detector with diversity.

3.2 Linear Detector

Here, we wish to determine linear detectors which are optimal in the minimum mean square sense.
Synchronous Systems: Eqn.(10) can be rewritten as

$$\mathbf{y} = \mathbf{H}_s \mathbf{b} + \mathbf{n}_s, \tag{17}$$

where $\mathbf{y} = \left[\mathbf{y}^{(1)} \cdots \mathbf{y}^{(L)}\right]^T$, $\mathbf{n}_s = \left[\mathbf{n}^{(1)} \cdots \mathbf{n}^{(L)}\right]^T$, and

$$\mathbf{H}_s = \begin{bmatrix} \mathbf{R}^{(1)} & 0 & \cdots & 0 \\ 0 & \mathbf{R}^{(2)} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{R}^{(L)} \end{bmatrix} \begin{bmatrix} \mathbf{E}^{(1)} \\ \mathbf{E}^{(2)} \\ \vdots \\ \mathbf{E}^{(L)} \end{bmatrix}.$$

The cross-covariance matrix of $\mathbf{n}_s$ is given by,

$$\mathrm{E}\left[\mathbf{n}_s \mathbf{n}_s^T\right] = \mathbf{Q}_s = \mathbf{diag}\left(\left[\sigma_1^2 \mathbf{R}^{(1)}, \cdots, \sigma_L^2 \mathbf{R}^{(L)}\right]\right). \tag{18}$$

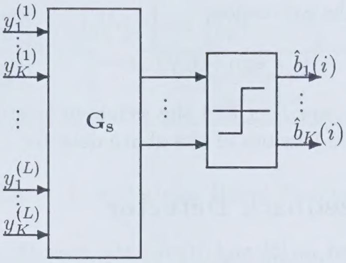


Figure 3: Linear Detector for Synchronous Systems

We want to find a linear transformation $\mathbf{G}_s : \mathbb{R}^{LK} \rightarrow \mathbb{R}^K$, such that the estimate of the vector $\mathbf{b}$ is $\hat{\mathbf{b}} = \mathbf{G}_s y$ and $\mathrm{E}\left[\left(\mathbf{b} - \hat{\mathbf{b}}\right)^{\mathbf{T}} \left(\mathbf{b} - \hat{\mathbf{b}}\right)\right]$ is minimized. In the absence of noise, this transformation is given by,

$$\mathbf{G}_s = \mathbf{H}_s{}^{\dagger}, \tag{19}$$

where † refers to the generalized matrix inverse. Parenthetically, we note that one would want to take a Moore-Penrose inverse, as it assures the uniqueness of the solution [7] [8](this is needed only if the linear independence assumption [3] of the signals is not valid - if it is valid, then we do not need more than one sensor to detect the symbols). In the presence of noise, it can be easily shown that the transformation is given by,

$$\mathbf{G}_s = \mathbf{H}_s^T \left(\mathbf{H}_s \mathbf{H}_s{}^T + \mathbf{Q}_s\right)^{\dagger}. \tag{20}$$

Note that $\mathbf{Q}_s$ would be positive definite if the signal set is linearly independent (not necessarily orthogonal) and the generalized inverse in the above equation becomes the regular matrix inverse. The symbols can then be estimated (since the alphabet is binary) as,

$$\hat{\mathbf{b}} = \mathbf{sgn}\left\{\mathbf{G}_s \mathbf{y}\right\}. \tag{21}$$

The structure of the detector is shown in Fig.3.
Asynchronous Systems: As in the case of the synchronous systems, we can rewrite eqn.(9) as,

$$\mathbf{y} = \mathbf{H}_a \mathbf{b} + \mathbf{n}_a \tag{22}$$

where $\mathbf{y} = \left[\mathbf{y}^{(1)} \cdots \mathbf{y}^{(L)}\right]^T$, $\mathbf{n}_a = \left[\mathbf{n}^{(1)} \cdots \mathbf{n}^{(L)}\right]^T$, and

$$\mathbf{H}_a = \begin{bmatrix} \mathcal{R}^{(1)} & 0 & \cdots & 0 \\ 0 & \mathcal{R}^{(2)} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathcal{R}^{(L)} \end{bmatrix} \begin{bmatrix} \mathcal{E}^{(1)} \\ \mathcal{E}^{(2)} \\ \vdots \\ \mathcal{E}^{(L)} \end{bmatrix},$$

with the cross-covariance matrix of $\mathbf{n}_a$ being given by,

$$\mathrm{E}\left[\mathbf{n}_a \mathbf{n}_a{}^T\right] \;=\; \mathbf{Q}_a \;=\; \mathbf{diag}\left(\left[\sigma_1^2 \mathcal{R}^{(1)}, \cdots, \sigma_L^2 \mathcal{R}^{(L)}\right]\right). \tag{23}$$

and the required linear transformation $\mathbf{G}_a : \mathbb{R}^{LNK}, \rightarrow \mathbb{R}^{NK}$ given by,

$$\mathbf{G}_a = \mathbf{H}_a{}^T \left(\mathbf{H}_a \mathbf{H}_a{}^T + \mathbf{Q}_a\right)^{-1}, \tag{24}$$

and the estimates are given by the expression,

$$\hat{\mathbf{b}} = \mathrm{sgn}\left\{\mathbf{G}_a \mathbf{y}\right\}. \tag{25}$$

We note here that the linear equalizer and the weighted least squares detector proposed in [5] are basically modified versions of the above detector.

3.3 Decision Based Feedback Detector

This detector has been described in [5] and [4] for the case of one sensor. Here, we generalize this scheme when L sensors are available. Suppose it is desired to estimate $b_k(i)$, and assume that all the other data symbols' estimates are available. Then, the estimate of $b_k(i)$ can be obtained by

$$\hat{b}_k(i) = \mathrm{sgn}\left\{\sum_{l=1}^{L} \frac{\sqrt{E_k^{(l)}(i)}}{\sigma_l^2} z_k^{(l)}(i)\right\}, \tag{26}$$

where $z_k^{(l)}(i)$ is the result obtained by subtracting out the estimated interference from the matched-filter output. This can be formally expressed as,
Synchronous Systems:

$$z_k^{(l)}(i) = y_k^{(l)}(i) - \sum_{\substack{n=1 \\ k \neq n}}^{K} \sqrt{E_n^{(l)}(i)}\hat{b}_n(i) R_{kn}^{(l)}(0). \tag{27}$$

Asynchronous Systems:

$$\begin{aligned} z_k^{(l)}(i) = {} & y_k^{(l)}(i) - \sum_{\substack{n=1 \\ k \neq n}}^{K} \sqrt{E_n^{(l)}(i)}\hat{b}_n(i) R_{kn}^{(l)}(0) \\ & - \sum_{n=1}^{K} \sqrt{E_n^{(l)}(i-1)}\hat{b}_n(i-1) R_{kn}^{(l)}(1) - \sum_{n=1}^{K} \sqrt{E_n^{(l)}(i+1)}\hat{b}_n(i+1) R_{kn}^{(l)}(-1). \end{aligned} \tag{28}$$

Thus the estimate of $b_k(i)$ depends on the estimates of other data symbols for that period and the immediately preceding and the following periods. Hence, some initial estimates should be made on these data symbols. Due to this dependence, the performance of this detector is primarily governed by the accuracies of these initial estimates. If these estimates are perfect, we can obtain optimal performance. Since this is not possible, the performance would be slightly inferior to the optimal detector. The initial estimates can be obtained by using any of the above detectors. The structure of a decision based feedback detector is shown in Fig.4. This detector can be thought of as a decision feedback equalizer adapted to a system with diversity.

4 Partial Tracking of the Signals

In general, it is neither necessary, nor practical, to have all the users tracked by all the sensors. The energies of some of the users would be so low that these signals can not contribute a great interference nor is their strength sufficiently strong for any meaningful

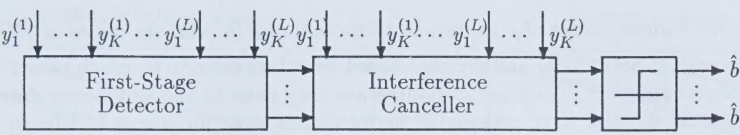


Figure 4: A Decision Based Feedback Detector

detection. Hence, we can assume that the contribution of these signals is negligible. This tracking can be handled much in the same way as the hand-off. Every base station will decide whether to receive data or not from a given user based on its SNR (some sort of threshold). If the SNR is higher than the threshold, the base station will receive data from the user even if there are some other base stations which are also receiving data from that receiver. If the SNR is less than the threshold it will not receive the data. This would greatly reduce the complexity of the detector as the bank of matched-filters now will have smaller number of matched filters as well as the amount of information that has to be sent to the switch would be reduced considerably.

For such a system, the detectors described in section 3 can be modified in a straightforward manner. For the corresponding linear MMSE detector, the number of equations to be solved is reduced to a great extent.

5 Error Probabilities: A Numerical Example

Expressions for a symbol error probability for the detectors discussed have been determined. For brevity, we omit these expressions. However, to evaluate the performance, we consider a numerical example. For a fair comparison, we let all the detectors use diversity for the purposes of detection.

Consider a synchronous system with three users and two sensors. We label the users as 1,2 and 3, and the sensors as 1 and 2. The first two users are tracked by the sensor 1, while the last two are tracked by sensor 2 (i.e., all the users are not tracked by all the users). Let,

$$\mathbf{R}^{(1)} = \begin{bmatrix} 1 & \rho_1 & 0 \\ \rho_1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{R}^{(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & \rho_2 \\ 0 & \rho_2 & 1 \end{bmatrix},$$

$$\mathbf{E}^{(1)} = \begin{bmatrix} E_1^{(1)} & 0 & 0 \\ 0 & E_2^{(1)} & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{E}^{(2)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & E_2^{(2)} & 0 \\ 0 & 0 & E_3^{(2)} \end{bmatrix}.$$

Furthermore, Let σ_1^2 and σ_2^2 be the two-sided power spectral density of the additive noise at the two sensors We assume them to be equal. We are interested in the performance of the detector with respect to user 2. Let $E_2^{(1)}$ and $E_2^{(2)}$, the energies of the second user at the two sensors, be equal. Further, let us assume that $E_1^{(1)} = E_3^{(2)}$ and $\rho_1 = \rho_2 = \rho$. We consider the cases when SNR1 $\left(= \frac{E_1^{(1)}}{2\sigma_1^2} \right)$ = SNR2 is 5dB as well as 10 dB. In Fig.5-10, we plot the average error probabilities of the conventional detector, the linear detector, the decision based detector, which employs the conventional detector to make its initial estimates, and a single-user detector (optimal detector used when there were no multi-user interference - see section 3.1) as a function of the signal-to-noise ratio SNR21

$\left(= \frac{E_2^{(1)}}{2\sigma_1^2}\right)$, for various values of ρ as well as a function of ρ for various values of SNR21. The performance of both linear and decision based detectors seem to be much better than the conventional detector. Their performances are very close to the single-user detector for ρ's as high as 0.3 and have reasonable performances even for $\rho = 0.5$. Clearly, the mitigation of the near-far effect is demonstrated by these results. Also, when there is a strong interference, the decision based feedback detector performs very well when the SNR21 is low. However, when SNR21 is high, it is outperformed by the linear detector. This is due to the fact that as SNR21 is increased, it causes significant interference to the other users' signals and hence the initial estimates made by the conventional receiver are not as reliable as when SNR21 is low.

6 Conclusions

In this paper, we have highlighted the diversity which is available in most cellular networks. We further developed sub-optimal receivers for this system using the concept of multiuser detection. By using the partial tracking strategy, it seems that the increase in complexity does not seem to be a big factor given the added advantage of diversity since the detectors developed have a complexity which is linear in the number of users, K as well as the degree of diversity L.

References

[1] S. Verdú. Minimum probability of error for asynchronous gaussian multiple-access channels. *IEEE Trans. Inform. Theory*, IT-32:85–96, Jan. 1986.

[2] R. Lupas and S. Verdú. Near-far resistance of multi-user detectors in asynchronous channels. *IEEE Trans. Commun.*, COM-38:496–508, April 1990.

[3] R.Lupas and S.Verdú. Linear multiuser detectors for synchronous code-division multiple-access channels. *IEEE Trans. Inf. Theory*, 35:123–136, Jan. 1989.

[4] M. K. Varanasi and B. Aazhang. Multistage detection in asynchronous code-divison multiple access communications. *IEEE Trans. Commun.*, COM-38:509–519, April 1990.

[5] Z. Xie, Robert T. Short, and C. K. Rushforth. Suboptimum coherent detection of direct sequence multiple access signals, 1989.

[6] L.F.Chang and J.C.-I.Chuang. Diversity selection using coding in a portable radio communications channel with frequency-selective fading. *IEEE J. Select. Areas. Commun.*, 7:89–98, Jan. 1989.

[7] R.A.Horn and C.R.Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

[8] S.L.Campbell and Jr C.D.Meyer. *Generalized Inverses of Linear Transformations.* Pitman Publishing Limited, London, 1979.

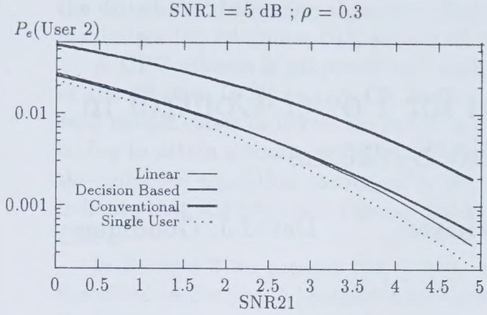

SNR1 = 5 dB ; $\rho = 0.3$

Figure 5:

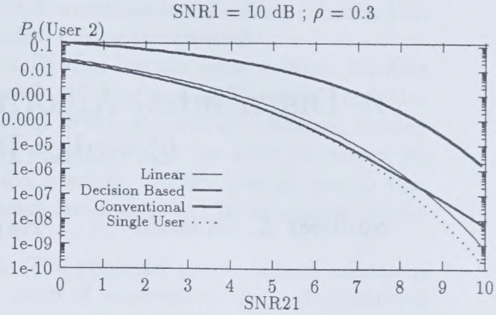

SNR1 = 10 dB ; $\rho = 0.3$

Figure 8:

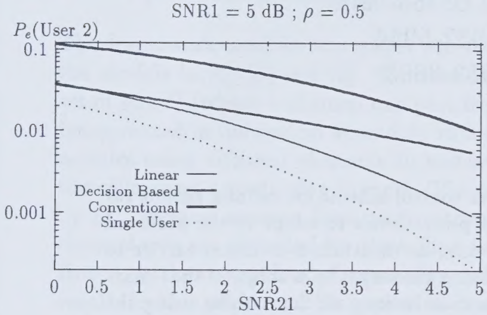

SNR1 = 5 dB ; $\rho = 0.5$

Figure 6:

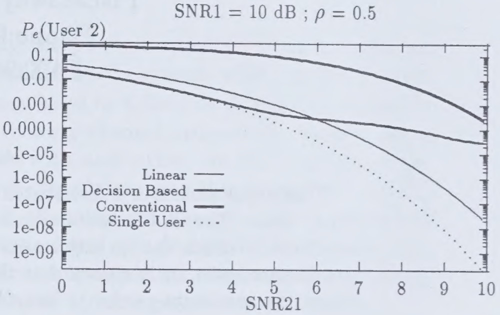

SNR1 = 10 dB ; $\rho = 0.5$

Figure 9:

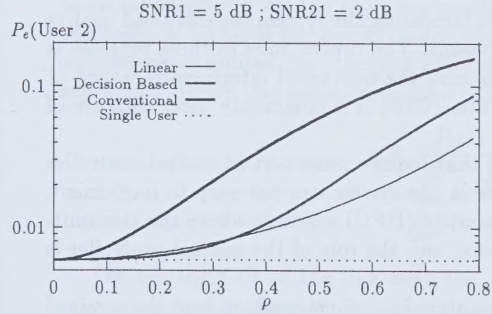

SNR1 = 5 dB ; SNR21 = 2 dB

Figure 7:

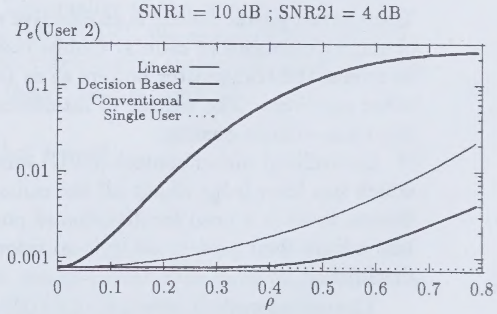

SNR1 = 10 dB ; SNR21 = 4 dB

Figure 10:

A Distributed Algorithm for Power Control in Cellular Radio Systems

Sudheer A. Grandhi Rajiv Vijayan David J. Goodman

Wireless Information Network Laboratory (WINLAB)
Rutgers, The State University of New Jersey
P.O. Box 909
Piscataway, NJ 08855-0909
Phone:908-932-5954
FAX:908-932-3693

Abstract

This paper describes a distributed power control scheme for cellular mobile radio systems. The mobile units adjust their power levels to adapt to the observed interference. Each mobile adjusts its power so as to attain a common carrier to interference ratio. It is shown that the scheme converges to a solution that maximizes the minimum carrier to interference ratio among all the mobiles using the channel.

1 Introduction

Transmitter power control is an effective way of increasing the system capacity and quality of communications in cellular mobile radio systems. The central idea in these schemes is to adjust the transmitter powers so as to minimize the cochannel interference caused at other receivers. The carrier to interference ratio (CIR) is a commonly used measure of the transmission quality.

Centralized power control (CPC) schemes that require some sort of central controller which has knowledge about all the radio links in the system are not easy to implement. Hence, there is a need for distributed power control (DPC) schemes where the transmitters adjust their powers using local information, and the role of the central controller is diminished, or even eliminated.

The initial work on power control ([1]-[3]) centered on schemes which kept the received power at a constant level. The advantage of this is that the dynamic range requirements for the receiver are smaller resulting in better adjacent channel protection. These schemes indicate an increase of system capacity by a factor roughly equal to two relative to the schemes with constant transmitted power. In [4], a CPC scheme that achieves the same CIR in all the radio links is proposed. This idea was further developed in [5], where an algorithm is presented that is optimal in the sense that it minimizes the outage probability. These results were used in [6] to derive an iterative DPC scheme called the distributed balancing algorithm (DBA), which converges to a limiting solution that maximizes the minimum CIR among all the mobiles using the channel.

A CPC scheme is proposed and analyzed in [7] that has the same optimal solution as the schemes in [5] and [6]. In the present work we propose a DPC scheme in which each mobile raises or lowers its power in proportion to the observed interference, thereby trying to attain a common CIR on all links. We prove that, like the DBA scheme of [6], the proposed algorithm converges to the maxmin solution. We also present results that indicate that the proposed scheme converges much faster than the DBA scheme at high CIR's.

In Section 2 we present the system model. The proposed power control scheme is described in Section 3. Section 4 includes the proof of convergence and a discussion of the factors affecting the rate of convergence. A numerical example is presented in Section 5 that compares the convergence of the proposed scheme with that of the DBA scheme.

2 System Model

The framework used in this paper for studying the power control problem is based on the models in [4], [5] and [6]. We consider a cellular radio system with a finite channel set of size N (where a channel can be a frequency band or a time slot), with the channels being reused in the system according to some arbitrary channel assignment scheme. Since mobiles using different channels do not interfere with each other, we will concentrate on the cells using a particular channel. The number of mobiles using this channel is denoted by M. We also assume that, as in current and proposed cellular systems, independent channel sets are allocated for the uplink (mobile-to-base) and downlink (base-to-mobile). Hence, without any loss of generality, we restrict our attention to the uplink. All the results in this paper can be applied to the downlink by changing the notation.

We let P_i be the power transmitted by the i^{th} mobile unit, which is communicating with the i^{th} base. The gain on the communication link between the i^{th} base and the j^{th} mobile is denoted by G_{ij}. Thus, the G_{ii}'s are the gain factors on the desired communication links, while the G_{ij}'s, for $i \neq j$, represent gain factors on the links that cause interference. All the G_{ij}'s are positive. The quality of the communication link is expressed in terms of the carrier-to-interference ratio (CIR). We assume that the system is interference limited, i.e., thermal noise and other extraneous sources of interference are neglected. Under this assumption, the CIR at the i^{th} base is given by

$$\gamma_i = \frac{P_i G_{ii}}{\sum\limits_{\substack{j=1 \\ j \neq i}}^{M} P_j G_{ij}} \, , \; 1 \leq i \leq M \, . \tag{1}$$

We use the CIR of the link with the worst quality, i.e., $\min\limits_{1 \leq i \leq M} \gamma_i$, as a performance measure to be maximized. As in [5], we define the maximum achievable CIR γ^* by

$$\gamma^* = \max_{\boldsymbol{P} \geq \mathbf{0}} \; \min_{1 \leq i \leq M} \gamma_i \, , \tag{2}$$

where $\boldsymbol{P}$ is the M-dimensional power vector whose i^{th} element is P_i. The notation $\boldsymbol{P} \geq \mathbf{0}$ denotes that the elements of $\boldsymbol{P}$ are nonnegative. A power vector $\boldsymbol{P}$ is said to be optimal if the resulting CIR's satisfy $\min\limits_{1 \leq i \leq M} \gamma_i = \gamma^*$. It has been shown in [5] that an optimal power vector exists, is unique up to scaling by a constant, and results in all links having the same CIR, i.e., $\gamma_i = \gamma^*$, $1 \leq i \leq M$.

3 The Proposed DPC scheme

In this section, we propose a DPC scheme in which the mobiles adjust their transmitter powers synchronously in discrete time steps. The power adjustment made by the i^{th} mobile at the n^{th} time instant is given by

$$P_i^{(n)} = c\,\frac{P_i^{(n-1)}}{\gamma_i^{(n-1)}},\ 1 \leq i \leq M\ ,\ n \geq 1\ , \tag{3}$$

where $P_i^{(n)}$ is the power transmitted by the i^{th} mobile at the n^{th} iteration, $\gamma_i^{(n)}$ is its resulting CIR, and c is some positive constant. The algorithm is initialized with an arbitrary positive initial vector $\boldsymbol{P}^{(o)}$. The two quantities involved in making the power control decision, viz. the current transmitted power and the current CIR, can both be measured by the mobile and its base without the involvement of a central controller. We also assume that the iterations are made quickly enough so that the link gains are constant during the time necessary for measurement and power adjustment.

The motivation for studying this algorithm is that all the mobiles can be considered to be striving to attain a common CIR. If, at the n^{th} step, the i^{th} mobile adjusts its power according to (3) while the other $M-1$ mobiles do not change their powers, then we would have $\gamma_i^{(n)} = c$. In that respect, c can be considered to be a "target" CIR that the mobiles are trying to attain. However, since all the mobiles adjust their powers at the same instant, the CIR's after making the adjustments will be different from c. In fact, we see that the CIR's are independent of the constant c, since changing c corresponds to scaling all the transmitter powers by the same factor, which does not affect the CIR. Note also that the CIR's will be unaffected if a different positive number $c^{(n-1)}$ is chosen at each iteration.

Equation (1) for the CIR at the i^{th} base can be rewritten as

$$\gamma_i = \frac{P_i}{\sum_{i=1}^{M} A_{ij} P_j}\ , \tag{4}$$

where the $M \times M$ matrix $\boldsymbol{A} = \{A_{ij}\}$ is defined by

$$A_{ij} = \begin{cases} \left(\dfrac{G_{ij}}{G_{ii}}\right) & \text{if } i \neq j \\ 0 & \text{if } i = j\ . \end{cases} \tag{5}$$

Thus, equation (3) can be rewritten as

$$P_i^{(n)} = c^{(n-1)} \sum_{j=1}^{M} A_{ij} P_j^{(n-1)},\ i = 1, \ldots, M\ ,\ n \geq 1\ , \tag{6}$$

or, in matrix notation,

$$\boldsymbol{P}^{(n)} = c^{(n-1)} \boldsymbol{A} \boldsymbol{P}^{(n-1)}\ ,\ n \geq 1\ . \tag{7}$$

Hence, we have

$$\boldsymbol{P}^{(n)} = \boldsymbol{A}^n \boldsymbol{P}^{(o)} \prod_{i=0}^{n-1} c^{(i)}. \tag{8}$$

4 Proof of Convergence

In this section, we shall show that the proposed DPC scheme converges to an optimal solution in the sense that

$$\lim_{n\to\infty} \gamma_i^{(n)} = \gamma^* \ , \ 1 \le i \le M \ . \tag{9}$$

In order to prove this result, we need to use some properties of the optimal solution. These were proved in [5] by applying the Perron-Frobenius theorem for strictly positive matrices to $\boldsymbol{Z} \overset{\text{def}}{=} \boldsymbol{A} + \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. For our purposes, it is more convenient to state these results in terms of the matrix $\boldsymbol{A}$ itself. We use the fact that, for $M \ge 3$, $\boldsymbol{A}$ is irreducible since $\boldsymbol{A}^2$ is strictly positive. This enables us to use the Perron-Frobenius theorem for nonnegative irreducible matrices ([8] - [11]), which we now state.

Theorem 1 (O. Perron and G. Frobenius): Let $\boldsymbol{A}$ be an $M \times M$ irreducible nonnegative matrix with eigenvalues $\{\lambda_i\}_{i=1}^{M}$. Then:
(a) $\boldsymbol{A}$ has a positive real eigenvalue λ^* with $\lambda^* = \max\{|\lambda_i|\}_{i=1}^{M}$.
(b)λ^* above has an associated eigenvector $\boldsymbol{P}^*$ with strictly positive entries.
(c)λ^* has algebraic multiplicity equal to 1.
(d)All eigenvalues λ of $\boldsymbol{A}$ other than λ^* satisfy $|\lambda| < |\lambda^*|$ if and only if there is a positive integer k with all entries of $\boldsymbol{A}^k$ strictly positive.
(e) The minimum real λ such that the inequality

$$\lambda \boldsymbol{P} \ge \boldsymbol{A}\boldsymbol{P} \tag{10}$$

has solutions for $\boldsymbol{P} \ge \mathbf{o}$ is $\lambda = \lambda^*$.
(f) The maximum real λ such that the inequality

$$\lambda \boldsymbol{P} \le \boldsymbol{A}\boldsymbol{P} \tag{11}$$

has solutions for $\boldsymbol{P} \ge \mathbf{o}$ is $\lambda = \lambda^*$.
(The notation $\boldsymbol{A} \ge \boldsymbol{B}$ is used to denote that each element in matrix $\boldsymbol{A}$ is greater than or equal to the corresponding element in matrix $\boldsymbol{B}$.)□

From Theorem 1, we obtain the following result, which is proved in [7].

Proposition 1: There exists a unique maximum achievable CIR γ^* given by

$$\gamma^* = \max_{\boldsymbol{P} \ge \mathbf{o}} \ \min_{1 \le i \le M} \{\gamma_i\} = \min_{\boldsymbol{P} \ge \mathbf{o}} \ \max_{1 \le i \le M} \{\gamma_i\} = \frac{1}{\lambda^*} \tag{12}$$

where λ^* is the largest real eigenvalue of the matrix $\boldsymbol{A}$. The power vector $\boldsymbol{P}^*$ that achieves γ^* is the eigenvector corresponding to λ^*.□

Using Theorem 1 and Proposition 1, we obtain the following result for the limiting behavior of the DPC scheme, which is analogous to the corresponding result for the DBA in [6].

Proposition 2: For a system with $M \geq 3$ that uses the DPC scheme with $c^{(n)}$, $n \geq 0$ chosen so that

$$\lim_{n\to\infty} (\lambda^*)^n \prod_{i=0}^{n} c^{(i)} < \infty ,$$

we have

$$\lim_{n\to\infty} \boldsymbol{P}^{(n)} = \boldsymbol{P}^* b \lim_{n\to\infty} (\lambda^*)^n \prod_{i=0}^{n} c^{(i)} , \tag{13}$$

and

$$\lim_{n\to\infty} \gamma_i^{(n)} = \gamma^*,\ i = 1, \ldots, M, \tag{14}$$

where b is a positive constant determined by $\boldsymbol{P}^{(o)}$.
Proof: Equation (8) represents the power method for finding the largest eigenvalue and the corresponding eigenvector of a matrix [13, pages 570-571]. Since, by Theorem 1 and Proposition 1, the largest eigenvalue λ^* of $\boldsymbol{A}$ and its corresponding eigenvector $\boldsymbol{P}^*$ are unique and give the maxmin solution, the result follows. □.

As noted earlier, the $c^{(n)}$'s do not affect the CIRs. However, they need to be chosen properly to prevent the transmitter powers from becoming too large or too small. A possible choice for $c^{(n)}$ is $1/\max\{P_i^{(n)}\}_{i=1}^{M}$, which normalizes the powers so that the largest power is equal to 1. This would of course necessitate some sort of communication between the base stations.

We now compare the proposed DPC scheme to the DBA scheme presented in [6]. In the DBA scheme, the iteration for the power levels is described by

$$P_i^{(n)} = c^{(n-1)} \left(P_i^{(n-1)} + \frac{P_i^{(n-1)}}{\gamma_i^{(n-1)}} \right),\ i = 1, \ldots, M \ ,\ n \geq 1 \ , \tag{15}$$

which can be written in matrix form as

$$\boldsymbol{P}^{(n)} = c^{(n-1)} \boldsymbol{Z} \boldsymbol{P}^{(n-1)} \ , \tag{16}$$

where $\boldsymbol{Z} = \boldsymbol{A} + \boldsymbol{I}$. Hence, we have

$$\boldsymbol{P}^{(n)} = \boldsymbol{Z}^n \boldsymbol{P}^{(o)} \prod_{i=0}^{n-1} c^{(i)} \ . \tag{17}$$

It has been shown in [6] that this scheme converges to the optimal solution.

The rate of convergence of the power method is determined by the ratios between the other eigenvalues and the largest one. The smaller these ratios, the faster the convergence. The rate of convergence of the DBA scheme depends on the eigenvalues of $\boldsymbol{Z}$. It is shown in [6] that when γ^* is large all the eigenvalues of $\boldsymbol{Z}$ are close to 1, thereby causing convergence to be slow. Since the eigenvalues of $\boldsymbol{A}$ are obtained by subtracting 1 from the corresponding eigenvalues of $\boldsymbol{Z}$, they will all be close to 0 when γ^* is large. This indicates that the proposed DPC scheme, the rate of convergence of which depends on the eigenvalues of $\boldsymbol{A}$, has the potential to offer faster convergence than the DBA scheme when γ^* is large.

5 Numerical Results

In this section, we present simulation results to compare the convergence of the proposed DPC scheme with that of the DBA scheme proposed in [6]. A one-dimensional cellular system with 50 cells is considered. There are 3 channels in the system, with each channel being reused in every third cell. There is one mobile in each cell, and mobile positions are distributed uniformly within each cell. The attenuation factors G_{ij} are given by

$$G_{ij} = d_{ij}^{-4} ,$$

where d_{ij} is the distance between the i^{th} base and the j^{th} mobile. The mobiles were frozen in their positions, and the power control schemes were applied. Both schemes were started with equal transitter power levels for all the mobiles. The choice for $c^{(n)}$ was $1/\max\{P_i^{(n)}\}_{i=1}^{M}$ in the DPC scheme and $1/|P^{(n)}|$ in the DBA scheme. The minimum CIR among the mobiles using a particular channel is plotted against the number of iterations in Figures 1, 2, and 3 for channels 1, 2, and 3, respectively, for a representative set of mobile positions. For this set of mobile positions, the maximum achievable CIR's for channels 1, 2, and 3 are 33.52 dB, 31.83 dB, and 34.02 dB, respectively. From the figures, we see that in each channel the proposed scheme converges much faster than the DBA scheme, with very few iterations being required for the minimum CIR to approach γ^*. However for lower values of γ^* close to say 10dB there is not much difference in the performance of the two schemes.

6 Conclusions

A distributed power control scheme that controls the transmitter powers to adjust the CIR to a common value for all the receivers has been presented. It is analyzed and shown to maximize the minimum of the CIR's and to also simultaneously minimize the maximum of the CIR's, of all the mobiles using a given channel.

Acknowledgement

We thank Jens Zander for making this work possible and for the valuable discussions we had with him.

References

[1] W. Tschirks, "Effects of transmission power control on the cochannel interference in cellular radio networks", Electrotechnik und Informationstechnik, Vol. 106, nr 5,1989.

[2] T. Fujii and M. Sakamoto,"Reduction of cochannel interference in cellular systems by intra-zone channel reassignment and adaptive transmitter power control", Proc. IEEE Veh. Tech. Conf., VTC-88,1988,pp. 668-672.

[3] T. Nagatsu, T. Tsuruhara, and M. Sakamoto, "Transmitter power control for cellular land mobile radio", Proc. IEEE Globecom 1983, pp. 1430-1434.

[4] R. W. Nettleton and H. Alavi, "Power control for spread-spectrum cellular mobile radio system", Proc. IEEE Veh. Tech. Conf., VTC-83, 1983, pp. 242-246.

[5] J. Zander, "Performance of optimum transmitter power control in cellular radio systems", *IEEE Transactions on Vehicular Technology*, vol. 41, no. 1, February 1992.

[6] J. Zander,"Distributed cochannel interference control in cellular radio systems", *IEEE Transactions on Vehicular Technology*, vol. 41, no. 3, August 1992.

[7] S. A. Grandhi, R. Vijayan, and D.J. Goodman, "Centralized Power Control", Technical Report WINLAB-TR-40, August 1992.

[8] B. Noble and J. Daniel, *Applied Linear Algebra.* third edition, pp. 375-376, Prentice-Hall, 1988.

[9] J. J. Hunter, *Mathematical Techniques of Applied Probability. Volume 1. Discrete Time Models: Basic Theory.* Academic Press, 1983.

[10] F. R. Gantmacher, *The Theory of Matrices.* volume 2, Chelsea Publishing Company, 1964.

[11] R. Bellman, *Introduction to Matrix Analysis.* McGraw-Hill, 1960.

[12] H. Minc, *Nonnegative Matrices.* John Wiley & sons, 1988.

[13] J. H. Wilkinson, *The Algebraic Eigenvalue Problem.* Clarendon Press: Oxford, 1988.

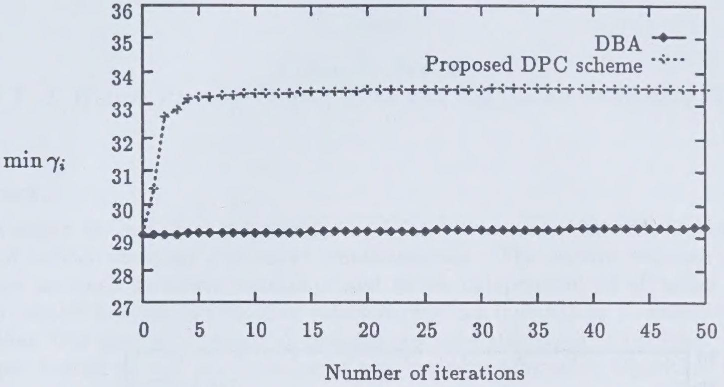


Figure 1: Convergence of the minimum CIR in channel 1.

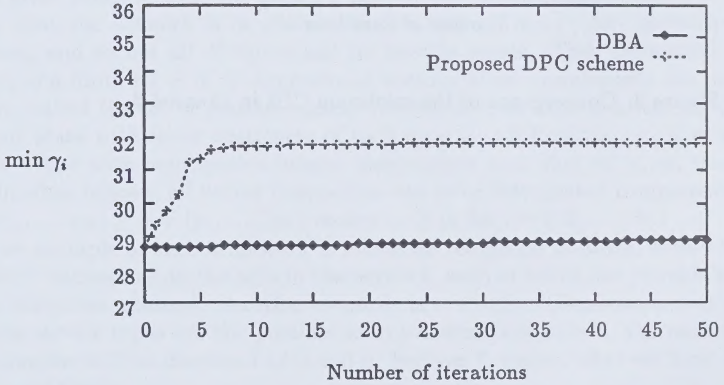


Figure 2: Convergence of the minimum CIR in channel 2.

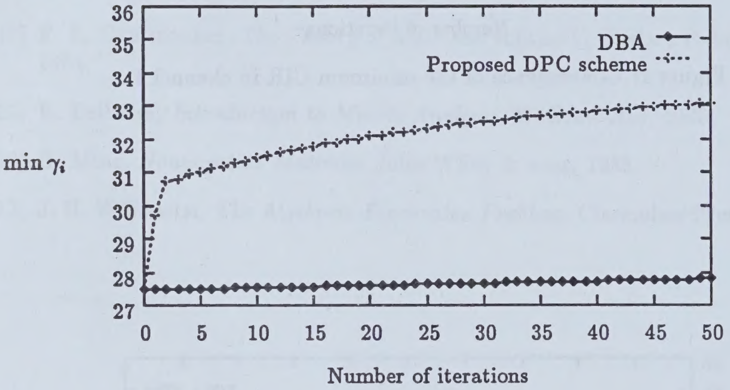


Figure 3: Convergence of the minimum CIR in channel 3.

Maximizing Marginal Revenue In Generalized Blocking Service Networks

Robert J. McEliece*
California Institute of Technology, 116-81, Pasadena, CA 91125
and
Kumar N. Sivarajan
IBM T. J. Watson Research Center, P. O. Box 704, Yorktown Heights, NY 10598

Abstract.

In this paper we introduce a general model for service networks which provide many kinds of service to many customers simultaneously. The service requests of a given type are assumed to arrive randomly and to be independent of all other types. A service request is either accepted or rejected (but not queued) by a *resource allocation algorithm.* Our goal is the study of optimal resource allocation algorithms. We obtain an upper bound on the performance of any resource allocation algorithm for such a network and show that this upper bound is, in a certain sense, asymptotically achievable. We present two specific applications of our general results — channel allocation in cellular communication networks and routing in wireline telephone networks.

1. Introduction. Statement of Results.

In this paper, we shall present a general model for service networks which provide many different types of service to many customers simultaneously. Our goal is the analysis and design of optimal resource allocation algorithms (i.e., algorithms which maximize marginal revenue as a function of customer demand) for such networks.

Consider then an abstract network capable of providing N different types of service. If at a given point in time the network has m_i customers of type i, for $i = 1, 2, \ldots N$, we say that the network is in *state* $m = (m_1, m_2, \ldots, m_N)$. Any network has finite resources, and so not all N-tuples can be feasible states. Thus associated with each network is a finite set S of N-dimensional vectors whose components are nonnegative integers, called the set of *feasible states.* We will assume that if a state m is feasible, then any state with fewer customers of each type is still feasible, i.e., if $m \in S$ and if m' is a vector with nonnegative integer components such that $m' \leq m$, then $m' \in S$, too. (In what follows, all vector inequalities are to be interpreted componentwise, e.g., $x = (x_1, \ldots, x_N) \geq y = (y_1, \ldots, y_N)$ means $x_i \geq y_i$ for $i = 1, 2, \ldots, N$.)

One example of such a network is a cellular telephone network, where the "types of service" correspond to the cells in the network, each of which can provide a customer with a telephone channel. Another example is a wireline telephone network. In this case, the service types are the possible source-destination pairs in the network. These two examples will be discussed in detail in Section 7, below, after we have developed our general theory.

We assume that a type i customer must pay b_i dollars per hour for the network's services, independent of all other customers. The vector $b = (b_1, \ldots, b_N)$ then may be called the *unit marginal revenue distribution*, because if the network is in state m

---

* McEliece's contribution to this paper was supported by a grant from GTE Laboratories, by AFOSR grant 91-0037, and by a grant from Pacific Bell.

the network's marginal revenues are mb^T dollars per hour. (See Section 8 for a brief discussion of how our results extend to more general marginal revenue functions.)

We call the network described by the set S and the vector b the *(S, b)-network.*

Next we describe the demand for services, which we measure in *Erlangs.* A service demand of ρ Erlangs represents an average of ρ potential customers for the network. We say *potential* since if the demand exceeds the network's resources, not all of the customers who desire service can be served. We assume that requests for service of type i are independent of service requests of all other types, and that the demand for type i service is r_i Erlangs. The vector $r = (r_1, \ldots, r_N)$ is then called the *demand vector.*

When a service request arrives at the network, we assume that the network administrator can do one of two things: accept the request, in which case the customer is serviced until his demand is satisfied, or block the request, in which case the customer is politely but firmly removed from further consideration. (Thus we do not allow the network to drop customers while they are being serviced, nor do we permit service queues.) The decision to accept or reject a service request is governed by a *resource allocation algorithm.* A resource allocation algorithm *must* reject a request if accepting it would move the network into an infeasible state, but *may* reject it even if it does not. The study of resource allocation algorithms is the central focus of this paper. We will measure the quality of a resource allocation algorithm by its *expected marginal revenue function* $T(r)$, which is defined to be the average value of the marginal revenue mb^T as a function of r, the demand vector, with S and b fixed.

Our main results deal not with *single* (S, b) networks of the type described above, but rather with *sequences* of such networks $(S_1, b), (S_2, b), \ldots$, where S_n represents the original network with "scale factor" n. For example, the sequence (S_n) might represent a sequence of cellular telephone networks, all with the same arrangement of cells, but with S_n having n available frequencies. Alternatively, (S_n) might represent a sequence of ordinary telephone networks, each with the same network topology, but with the trunks in S_n having capacity proportional to n.

To describe the network scaling mathematically, we assume there is an underlying "proto-set" $\mathcal{S}$, which will serve as an "unscaled model" model for the feasible regions S_n. We require the set $\mathcal{S}$ to be a convex subset of R^N with the further property that if $s \in \mathcal{S}$, and $0 \le s' \le s$, then $s' \in \mathcal{S}$, too. In the sequence (S_n) of sets of feasible states, we shall call S_n, which is supposed to represent the proto-set $\mathcal{S}$ at "scale factor" n, the set of *n-feasible states.* We require the sequence (S_n) to be related to the proto-set $\mathcal{S}$ by the following two conditions:

$$S_n/n \subseteq \mathcal{S} \qquad \text{for all } n = 1, 2, \ldots. \tag{1.1}$$

$$\lim_{n\to\infty} S_n/n \supseteq \mathcal{S}. \tag{1.2}$$

(What (1.1) means is that if $s_n \in S_n$, then $s_n/n \in \mathcal{S}$. What (1.2) means is that for every point $s \in \mathcal{S}$, there is a sequence (s_n), with $s_n \in S_n$, such that $s_n/n \to s$.)

With the proto-set $\mathcal{S}$ and the sequence (S_n, b) of scaled networks fixed, we denote by $T_n(r)$ the maximum possible marginal revenue for the (S_n, b) network, as a function of the demand vector r. Our main concern in this paper is the relationship of the proto-set $\mathcal{S}$ to the behavior of the functions $T_n(r)$.

Here is a summary of the results in the paper. In Section 2 we shall define and briefly study a certain function $T_{\mathcal{S},b}(r)$, which is monotonic nondecreasing in r, continuous, and convex $\cap$. In Section 3 we shall see that

$$\frac{1}{n}T_n(nr) \le T_{\mathcal{S},b}(r) \qquad \text{for } n = 1, 2, \ldots. \tag{1.3}$$

In Section 4, on the other hand, we will see that

$$\lim_{n\to\infty} \frac{1}{n} T_n(nr) = T_{\mathcal{S},b}(r). \tag{1.4}$$

In view of (1.3) and (1.4), we call the function $T_{\mathcal{S},b}(r)$ the *universal marginal revenue function* for the sequence (S_n, b).

In Section 5, we analyze the performance of a particular resource allocation algorithm that is easily described — the greedy resource allocation algorithm. In Section 6, we will see that if the proto-set $\mathcal{S}$ is a convex polytope, the function $T_{\mathcal{S},b}(r)$ can be computed by linear programming. In Section 7, we will give two applications of the general theory, one to cellular telephone networks (thereby reproducing the results in [5]) and one to a classic telephone routing problem ([3], section 7.5). Finally, in Section 8, we will briefly discuss how our results extend to more general revenue functions.

2. The function $T_{\mathcal{S},b}(r)$.

We begin with the definition of the function $T_{\mathcal{S},b}(r)$, whose significance we outlined in Section 1. $T_{\mathcal{S},b}(r)$ depends on the demand vector r through an auxiliary set $U(r)$, which is defined as follows.

$$U(r) = \{(u_1, \ldots, u_N) : u_i \le r_i \text{ for } i = 1, 2, \ldots, N.\} \tag{2.1}$$

Here is the definition of $T_{\mathcal{S},b}(r)$.

$$T_{\mathcal{S},b}(r) = \max\{sb^T : s \in \mathcal{S} \cap U(r)\}. \tag{2.2}$$

The most important mathematical properties of the function $T_{\mathcal{S},b}(r)$ are described in the following theorem.

Theorem 2.1. *The function $T_{\mathcal{S},b}(r)$ is nondecreasing, continuous, and convex $\cap$. Furthermore, $T_{\mathcal{S},b}(r) = rb^T$ if $r \in \mathcal{S}$, and $T_{\mathcal{S},b}(r) = \max\{sb^T : s \in \mathcal{S}\} = \beta$ independent of r, if $U(r)$ contains at least one point $s \in \mathcal{S}$ such that $sb^T = \beta$. Thus in particular, there exists a vector r_0 such that $T_{\mathcal{S},b}(r) = \beta$ for all $r \ge r_0$.*

Proof: Since the set over which the maximum in (2.2) occurs, viz. $\mathcal{S} \cap U(r)$, is increasing in r, it follows immediately that $T_{\mathcal{S},b}(r)$ is nondecreasing. The continuity of $T_{\mathcal{S},b}(r)$ can be proved in a routine manner, by appealing directly to the definition (2.2). Alternatively, continuity follows from convexity ([2], Section 1.5), which we now establish.

To prove that $T_{\mathcal{S},b}(r)$ is convex $\cap$, let r_1 and r_2 be arbitrary demand vectors, and let p and q be positive real numbers such that $p + q = 1$. Let $s_1 \in \mathcal{S} \cap U(r_1)$ achieve $T_{\mathcal{S},b}(r_1)$, i.e., $s_1 b^T = \max\{sb^T : s \in \mathcal{S} \cap U(r_1)\}$, and similarly let $s_2 \in \mathcal{S} \cap U(r_2)$ achieve $T_{\mathcal{S},b}(r_2)$. Now let $s = ps_1 + qs_2$. Then we have $s \in \mathcal{S} \cap U(pr_1 + qr_2)$, and

$$sb^T = (ps_1 + qs_2)b^T = ps_1 b^T + qs_2 b^T = p T_{\mathcal{S},b}(r_1) + q T_{\mathcal{S},b}(r_2). \tag{2.3}$$

From (2.3), it immediately follows from the definition (2.2) that

$$T_{\mathcal{S},b}(pr_1 + qr_2) \ge p T_{\mathcal{S},b}(r_1) + q T_{\mathcal{S},b}(r_2),$$

i.e., $T_{\mathcal{S},b}(r)$ is convex $\cap$.

Next, suppose that $r \in \mathcal{S}$. This is the same as saying $U(r) \subseteq \mathcal{S}$, and so

$$T_{\mathcal{S},b}(r) = \max\{sb^T : 0 \le s \le r\}. \tag{2.4}$$

Since the components of b are nonnegative, it follows that the maximum in (2.4) occurs when $s = r$, which shows that $T_{\mathcal{S},b}(r) = rb^T$ if $r \in \mathcal{S}$.

Finally suppose that $s_0 \in U(r)$, where $s_0 b^T = \beta = \max\{sb^T : s \in \mathcal{S}\}$. In this case, the definition (2.2) immediately implies that $T_{\mathcal{S},b}(r) = \beta$. ∎

3. $T_{\mathcal{S},b}(r)$ as an upper bound on system performance.

Let A be a resource allocation algorithm for the (S_n, b)-network. Then if the demand vector is nr, the state m is a random N-dimensional vector, which must satisfy the following two conditions.

$$E(m) \le nr \tag{3.1}$$

$$m \in S_n. \tag{3.2}$$

What (3.1) says is that the average number of customers can't exceed the average demand for service. (3.2) says that the network state must always be n-feasible. We define the marginal revenue function $T_A(nr)$ for the algorithm A as the expectation of the marginal revenue mb^T, when the demand is nr.

Theorem 3.1. *For any resource allocation algorithm A for the (S_n, b)-network, the corresponding marginal revenue function $T_A(nr)$ must satisfy*

$$\frac{1}{n}T_A(nr) \le T_{\mathcal{S},b}(r) \qquad \textit{for all } r \ge 0. \tag{3.3}$$

Proof: We have, by definition, and by the linearity of the expectation operator, $T_A(r) = E(mb^T) = E(m)b^T$. Since $m \in S_n \subseteq n\mathcal{S}$ (by (1.1)) and $n\mathcal{S}$ is convex, $E(m) \in n\mathcal{S}$. Combined with (3.1), we have $E(m) \in n\mathcal{S} \cap U(nr)$, so that

$$\begin{aligned} T_A(r) = E(m)b^T &\le \max\{sb^T : s \in n\mathcal{S} \cap U(nr)\} \\ &= n \max\{sb^T : s \in \mathcal{S} \cap U(r)\} \\ &= n\, T_{\mathcal{S},b}(r) \quad \text{(by (2.2))}, \end{aligned}$$

which completes the proof of Theorem 3.1. ∎

4. $T_{\mathcal{S},b}(r)$ is achievable for large scale factors.

To motivate the results in this section, we begin by considering a sequence (S_n, b) of *one-dimensional* networks, with $S_n = [0, s_n]$, where $s_n/n \to s$ and $b = 1$. The proto-set in this case is simply the closed interval $[0, s]$, and a simple calculation shows that for this case, the function $T_{\mathcal{S},b}(r)$ equals $\min(r, s)$. The obvious resource allocation algorithm for the simple network (S_n, b) is to accept a service request if and only if the resulting total number of customers is $\le s_n$. (In Section 5, we will briefly study a generalization of this "obvious" resource allocation algorithm.) If the intensity of the offered traffic is r_n Erlangs, with $r_n/n \to r$, we denote by $\tau(r_n, s_n)$ the corresponding marginal revenue, which in this case is simply the expectation of the number of customers being served. According to Theorem 3.1, we then have

$$\frac{\tau(r_n, s_n)}{n} \le \min(r, s), \qquad \text{for } n = 1, 2, \ldots. \tag{4.1}$$

In this section, we shall make the axiomatic assumption that the traffic model, which is independent of the network being studied and which we have so far left largely unspecified, satisfies

$$\lim_{n\to\infty} \frac{\tau(r_n, s_n)}{n} = \min(r, s), \tag{4.2}$$

for any pair of sequences r_n and s_n with $r_n/n \to r$ and $s_n/n \to s$. We call the property represented by (4.2) the "asymptotic traffic property" (ATP). It says that the bound in Theorem 3.1 is asymptotically tight for one-dimensional networks. Alternatively, it says that if the average demand is r_n and $r < s$, asymptotically the fraction of customers who are blocked is zero, whereas if $r > s$, then asymptotically the network is saturated. It is thus a kind of law of large numbers for the demand statistics. Many common demand models satisfy this property, including the standard Poisson arrivals, as was shown in [5].

Our main result is that if (4.2) holds, i.e., if the bound in Theorem 3.1 is asymptotically tight in the one-dimensional case, then Theorem 3.1 is asymptotically tight in general.

Theorem 4.1. *Assume the traffic model satisfies the ATP as described in (4.2). Then if $T_n(nr)$ denotes the largest possible average marginal revenue produced by any resource allocation algorithm for the (S_n, b)-network when the demand vector is nr, then*

$$\lim_{n\to\infty} \frac{1}{n} T_n(nr) = T_{\mathcal{S},b}(r). \tag{4.3}$$

Proof: By Theorem 3.1, we know that

$$\lim_{n\to\infty} \frac{1}{n} T_n(nr) \leq T_{\mathcal{S},b}(r). \tag{4.4}$$

It remains to show the opposite inequality. Thus with r fixed, choose $s \in \mathcal{S} \cap U(r)$ arbitrarily, and (using (1.2)) select a sequence $(s_n \in S_n)$ such that

$$\lim_{n\to\infty} \frac{s_n}{n} = s. \tag{4.5}$$

Now define the "fixed resource allocation algorithm of type s_n," or more briefly, the s_n-fixed algorithm, for the (S_n, b)-network as follows: A service request is accepted if and only if accepting it will move the system into a state $m' \in U(s_n)$, i.e., if and only if $m'_i \leq s^i_n$ for $i = 1, 2, \ldots, N$ where the m'_i are the components of m' and the s^i_n are the components of s_n. Since we have assumed that the demand for the various service types are independent, the components m_i of the state vector m are independent random variables, and so under the operation of the s_n-fixed algorithm the (S_n, b)-network decomposes into N independent sequences of one-dimensional networks of the type considered above. The ith network in this sequence is (S^i_n, b_i), where $S^i_n = [0, s^i_n]$ and the b_i are the components of b. We therefore have $E(m_i) = \tau(nr_i, s^i_n)$ for $i = 1, 2, \ldots, N$, and so the marginal revenue function for the s_n-fixed algorithm, which we temporarily denote by $T^*_n(nr)$, satisfies

$$T^*_n(nr) = E(mb^T) = \sum_{i=1}^{N} E(m_i)b_i = \sum_{i=1}^{N} \tau(nr_i, s^i_n)b_i. \tag{4.6}$$

If we divide (4.6) by n, and take the limit as $n \to \infty$, using the ATP (4.2), we obtain

$$\begin{aligned}
\lim_{n\to\infty} \frac{1}{n} T^*_n(nr) &= \lim_{n\to\infty} \sum_{i=1}^{N} \frac{\tau(nr_i, s^i_n)}{n} b_i \\
&= \sum_{i=1}^{N} \min(r_i, s_i)b_i. \\
&= \sum_{i=1}^{N} s_i b_i \quad \text{(since } s \in U(r)\text{)} \\
&= sb^T. \qquad (4.7)
\end{aligned}$$

But since the result in (4.7) holds for every $s \in \mathcal{S} \cap U(r)$, we may choose the $s = s(r)$ that maximizes it, i.e., the one for which $s(r)b^T = T_{\mathcal{S},b}(r)$. For this particular choice of s, we have from (4.7)

$$\lim_{n\to\infty} \frac{1}{n} T_n^*(nr) = T_{\mathcal{S},b}(r). \tag{4.8}$$

which, combined with (4.3), completes the proof. ∎

5. The greedy resource allocation algorithm.

Theorem 4.1 shows what level of performance can be achieved by the *best possible* resource allocation algorithms for a given sequence of networks. In fact, the proof shows that a specific class of algorithms, the so-called "fixed resource allocation algorithms," asymptotically achieve optimal performance. However, a closer look at the proof of Theorem 4.1 shows that in order to achieve the maximum value $T_{\mathcal{S},b}(r)$, a fixed algorithm must "know" the value of the demand vector r. In many situations, this value may be unknown, or changing, in which case fixed resource allocation algorithms are not of much practical value. In an attempt to remedy this weakness in our proof, in this next section we will briefly consider a class of "dynamic" resource allocation algorithms, the so-called "greedy" resource allocation algorithms, which automatically adjust themselves to changing values of r, and compare their performance to the optimal value promised by Theorem 4.1.

By definition, the "greedy" resource allocation algorithm accepts a service request if and only if the resulting state is feasible, i.e., it never rejects a service request unless it is forced to. Somewhat surprisingly, it is possible to give an explicit formula for the asymptotic performance of the greedy resource allocation algorithm. In this section we will give this formula without proof. For details, we refer to [4] and [6].

For a given value of r, let $s_g = (s_g^1, \ldots, s_g^N) = s_g(r)$ be the unique element of $\mathcal{S}$ that maximizes the function

$$\sum_{i=1}^{N} s_g^i \left(1 + \log \frac{r_i}{s_g^i}\right).$$

Theorem 5.1. *If the arrival process of service requests is Poisson and $T_n^G(nr)$ denotes the marginal revenue function for the greedy resource allocation algorithm for the (S_n, b)-network,*

$$T_G(r) = \lim_{n\to\infty} \frac{1}{n} T_n^G(nr) = s_g b^T, \qquad \textit{for all } r \geq 0. \tag{5.1}$$

6. When the proto-set $\mathcal{S}$ is described by linear inequalities.

In many applications, the set of n-feasible states, and the underlying proto-set $\mathcal{S}$ can be described by linear inequalities, as follows. There are fixed rational matrices $\mathcal{A}$ (dimensions $N \times K$) and $\mathcal{B}$ (dimensions $L \times K$) and a given rational vector $\mathcal{C}$ (dimension $1 \times K$). The proto-set $\mathcal{S}$ is defined to be the set of all non-negative *real* vectors $s \in R^N$ such that there exists a non-negative *real* vector $t \in R^L$ for which

$$s\mathcal{A} + t\mathcal{B} \leq \mathcal{C}. \tag{6.1}$$

Similarly, the set S_n of n-feasible vectors is defined as the set of all non-negative *integer* vectors m such that there exists a non-negative *integer* vector $l \in R^L$ for which

$$m\mathcal{A} + l\mathcal{B} \leq n\mathcal{C}. \tag{6.2}$$

With these definitions, it is relatively easy to show that (1.1) and (1.2) are satisfied. Also, when the set $\mathcal{S}$ is thus described by linear equalities, so is the set $\mathcal{S} \cap U(r)$ and since the objective function sb^T is linear, for a fixed r, $T_{\mathcal{S},b}(r)$ can be computed by *linear programming.* In the next section we will give some examples of applications where the set $\mathcal{S}$ is described by linear inequalities.

7. Applications.

We consider two applications — channel assignment in cellular communication networks and routing in wireline (telephone) networks. In both cases, as we shall see, the corresponding service networks can be described by linear inequalities of the general kind discussed in section 6.

Cellular Communication Networks. We consider cellular networks of the kind studied in [5], with n representing the number of available channels, and N representing the number of cells. The different types of service requests correspond to calls in different cells. Let the $N \times M$ matrix A denote the vertex-maximal independent set of the underlying hypergraph describing the reuse constraints. A state m is n-feasible iff there exists an integer vector $X \geq 0$ such that $X_1 + X_2 + \cdots + X_M \leq n$ and $XA^T \geq m$. Therefore, in the notation of (6.1) and (6.2), we can take $K = N + 1$, $L = M$, and

$$\mathcal{A} = (\, I_N \quad 0_N^T \,), \qquad \mathcal{B} = (\, -A^T \quad 1_M^T \,), \qquad \mathcal{C} = (0_N, 1).$$

(0_N denotes the all-zero N-vector, 1_M the all-one M-vector and I_N the $N \times N$ identity matrix.) With this start, it is now a routine matter to use the results in sections 2-6 of this paper to reproduce the results in [5]. Indeed, we arrived at the results described in this paper by generalizing those in [5].

Routing in Wireline Networks. Consider a telephone network with K nodes and M (undirected) links which interconnect these nodes in some fashion. Call requests arrive at random at the nodes, are destined for some other (random) node and have a random duration. We will assume that all calls are two-way (as is the case in telephony). The maximum number of (two-way) calls that can be carried on link j is c_j, i.e., the capacity of the link j is c_j *channels.* A routing algorithm for such a network must find a path between the source node at which the call arrives and its destination node and assign one channel in each of the links on this path. This situation fits the model of this paper, if we interpret the channels on the links as the resources, and the routing algorithm as the resource allocation algorithm. This routing problem is a special case of one considered in ([3], section 7.5), in which we maximize the "expected rate of revenue in the network" ([3], p. 311, last paragraph).

Suppose there are $N(\leq \binom{K}{2})$ source-destination (s-d) pairs in the network — these correspond to the different types of service requests. There will, in general, be several paths between the same s-d pair in the network that can be chosen by the routing algorithm (provided at least one channel is available on each of the links in the path). Let there be a total of P paths in the network. Let $A = (a_{ij})$ be the $P \times N$ path-s-d-pair incidence matrix, i.e.,

$$a_{ij} = \begin{cases} 1, & \text{if path } i \text{ is between s-d pair } j, \\ 0, & \text{otherwise.} \end{cases} \tag{7.1}$$

Let $B = (b_{ij})$ be the $P \times M$ path-link incidence matrix of the network, i.e.,

$$b_{ij} = \begin{cases} 1, & \text{if link } j \text{ is on path } i, \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

Let $c = (c_1, c_2, \ldots, c_M)$ be the vector of link capacities. A state $m = (m_1, \ldots, m_N)$ is feasible iff there exists a non-negative (integer) vector $l = (l_1, l_2, \ldots, l_P)$ (of path flows) such that

$$m \leq lA \qquad \text{and } lB \leq c. \tag{7.3}$$

The constraint $m \leq lA$ states that the total number of channels assigned to s-d pair i is at least m_i whereas the constraint $lB \leq c$ states that none of the link capacities are violated. In the notation of (6.1) and (6.2), we can take $K = N + M$, $L = P$, and

$$\mathcal{A} = (\, I_N \quad 0_{N \times M} \,), \qquad \mathcal{B} = (\, -A \quad B \,), \qquad \mathcal{C} = (0_N, c). \tag{7.4}$$

($0_{N \times M}$ denotes the $N \times M$ matrix of all zeroes.)

To illustrate, consider the simple 3-node network shown in Figure 1. In this example, there is exactly one path between each s-d pair, and the matrices A and B of (7.1) and (7.2) are

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and } B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

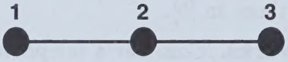


Figure 1. A Simple 3-Node Network.

Let $c = (1, 1)$ and $b = (1, 1, 1)$, which means that the links capacities are all equal, and the "marginal revenue" equals the expected number of calls in progress, i.e., the carried traffic. Let $r_1 = r_2 = r_3$ (uniform traffic) and let $|r| = r_1 + r_2 + r_3 = 3r_1$. For this network, a simple calculation using (2.2) shows that

$$T_{S,b}(r) = \min(|r|, 1 + |r|/3, 2).$$

For the same example, consider the performance of the greedy routing algorithm corresponding to the greedy resource allocation algorithm of Section 5. In this example, the greedy resource allocation algorithm routes a call on the unique path between the s-d pair if a channel is available on every link in that path, and blocks the request otherwise. Using the results in [4], we can show that,

$$T_G(r) = \min \left\{ |r|, 1 + \frac{|r|}{6} \left(\sqrt{1 + \frac{12}{|r|}} - 1 \right) \right\}.$$

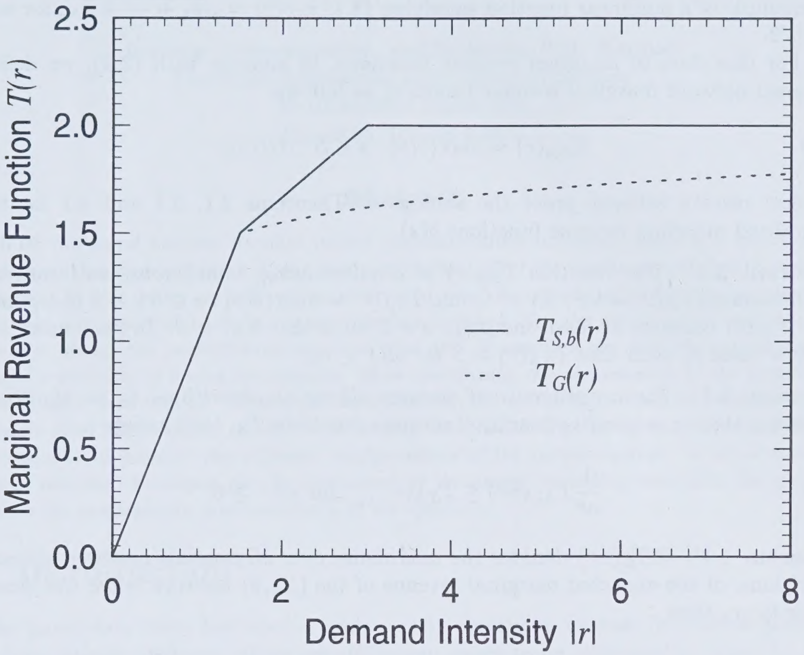


Figure 2. The functions $T_{S,b}(r)$ and $T_G(r)$ for the 3-node Network of Figure 1.

This is plotted along with $T_{S,b}(r)$ in Figure 2. It is gratifying that this simply-described (though possibly difficult to implement) algorithm asymptotically tracks the limiting function $T_{S,b}(r)$ for $0 \leq r \leq 3/2$. On the other hand, it is disappointing that for $r \geq 3/2$, the performance is so poor. We plan to study this interesting phenomenon further.

A slightly different routing problem is considered in [4]. Here, call requests arriving at the source request a *particular route* through the network to the destination (and not just the destination). This is a special case of our general service model where each route in the model of [4] represents a service type in our general model.

8. More general marginal revenue functions.

In Section 1, we defined the marginal revenue function at state $s = (s_1, \ldots, s_N)$ as sb^T where b was a fixed vector (the unit marginal revenue distribution). Thus the marginal revenue, which we denote by $b(s)$, is a *linear function* of the state s. In some situations, one can imagine a *nonlinear* marginal revenue function, perhaps because the network may realize certain economies of scale at high volume.

In any case, we can generalize our results to a large class of marginal revenue functions, viz., continuous, monotone nondecreasing, convex $\cap$ functions $b(s)$ which satisfy the following homogeneity property:

$$(8.1) \qquad b(ns) = n^p b(s), \qquad \text{for all } n = 1, 2, \ldots.$$

where $p > 0$. The function n^p indicates how the marginal revenue function "scales" with the network. If the marginal revenue function is linear, as before, $p = 1$ in (8.1). An example of a nonlinear function satisfying (8.1) is $b(s) = \sqrt{s_1 + \cdots + s_N}$, for which $p = 1/2$.

For this class of marginal revenue functions, in analogy with (2.2), we define a universal network marginal revenue function, as follows.

$$T_{\mathcal{S},b}(r) = \max\{b(s) : s \in \mathcal{S} \cap U(r)\}. \tag{8.2}$$

We now restate without proof the analogs of Theorems 2.1, 3.1 and 4.1 for these generalized marginal revenue functions $b(s)$.

Theorem 2.1′. *The function $T_{\mathcal{S},b}(r)$ is nondecreasing, continuous, and convex $\cap$. Furthermore, $T_{\mathcal{S},b}(r) = b(r)$ if $r \in \mathcal{S}$, and $T_{\mathcal{S},b}(r) = \max\{b(s) : s \in \mathcal{S}\} = \beta$ independent of r, if $U(r)$ contains at least one point $s \in \mathcal{S}$ such that $b(s) = \beta$. In particular, there exists a value r_0 such that $T_{\mathcal{S},b}(r) = \beta$ for all $r \geq r_0$.*

Theorem 3.1′. *For any sequence of resource allocation algorithms A_n for the (S_n, b)-networks, the corresponding marginal revenue functions $T_{A_n}(nr)$ satisfy*

$$\frac{1}{n^p} T_{A_n}(nr) \leq T_{\mathcal{S},b}(r) \qquad \textit{for all } r \geq 0. \tag{8.3}$$

Theorem 4.1′. *If $T_n(nr)$ denotes the maximum, over all possible resource allocation algorithms, of the expected marginal revenue of the (S_n, b)-network when the demand vector is nr, then*

$$\lim_{n \to \infty} \frac{1}{n^p} T_n(nr) = T_{\mathcal{S},b}(r). \tag{8.4}$$

References.

[1] Feller, W., *An Introduction to Probability Theory and Its Applications*, vol. 1, 3rd. ed. New York: Wiley, 1968.

[2] Fleming, W. H., *Functions of Several Variables.* Reading, Mass.: Addison-Wesley, 1965.

[3] Girard, A., *Routing and Dimensioning in Circuit-Switched Networks.* Reading, Mass.: Addison-Wesley, 1990.

[4] Kelly, F. P., "Blocking Probabilities in Large Circuit-Switched Networks," *Adv. in Appl. Prob.*, vol. 18, pp. 473–505, 1986.

[5] McEliece, R. J., and K. N. Sivarajan, "Performance Limits for Channelized Cellular Telephone Systems," submitted to *IEEE Trans. Inform. Theory.* (Earlier versions appeared in *Proc. 1990 Allerton Conference on Communication, Control and Computing*, pp. 869–880 and *Proc. Third IEE Conference on Telecomm.*, Edinburgh, pp. 360–365.)

[6] McEliece, R. J., and K. N. Sivarajan, "Heavy Traffic Performance of a Class of Channel Assignment Algorithms," Proc. 3rd Int. Symp. Personal, Indoor, and Mobile Radio Comm. (Boston, Oct. 1992), in press.

Performance Evaluation of Indoor Microcellular Systems

Ioannis Argyropoulos and **Srikanta P.R. Kumar**
Electrical Engineering and Computer Science
Northwestern University
Evanston, Illinois 60208

Abstract

With the trend towards wireless indoor communication networks, there is a growing demand for understanding of the parameters that affect the performance of such systems. In this paper we attempt to determine what the effects of various parameters are in a microcellular wireless communication network. For the indoor wireless system under study, the performance measures that will be assessed are blocking probability and probability of forced termination. More specifically, the performance of the system is studied with respect to radios per base station, call set up algorithm, distribution of radio fixed parts, signal strength and propagation model. The study is concluded with comparisons between the different configurations of the indoor system. A simulation tool has been developed for the evaluation of the above, providing invaluable insights into the performance and sensitivity of the system.

1 Introduction

In the past years there has been a continuously increasing interest in wireless personal communications. However, there are still many issues to be addressed in order for such systems to become commercially available, such as spectrum allocation, traffic capacity and channel structure. Of particular interest are the indoor office pico-cellular systems, which seem more technologically feasible than large area microcellular systems. The performance and understanding of such systems is however still limited, by the lack of an analytical model. There are many published studies of such systems through simulations [1],[2]. This paper presents the results from such an approach of an indoor micro (or "pico") cellular system. The issues that were addressed were the optimal number of radios/Base station, call set-up algorithm and capability of the portables to control the signal power. The effect of a certain traffic range on the probabilities of blocking and forced termination of the system, the sensitivity of the results to the propagation model, and the distribution of the radio fixed parts are also addressed.

2 Model

The system under study consists of a three-storey office building. The dimensions of each floor are 82×102 m. and the spacing between floors is 4 m. There are 14 Base Stations on each floor, placed as shown in figure 1. Portables are distributed uniformly on all three floors. Calls can be originated anywhere in the building, which is equivalent to the portables moving freely when they are not active. Once a call is established, it is assumed that the portable does not change its position. The configuration used for the system under study is the British specification for second generation wireless networks, namely CT2. It consists of 40 frequencies, having one time slot each [3]. The main reason

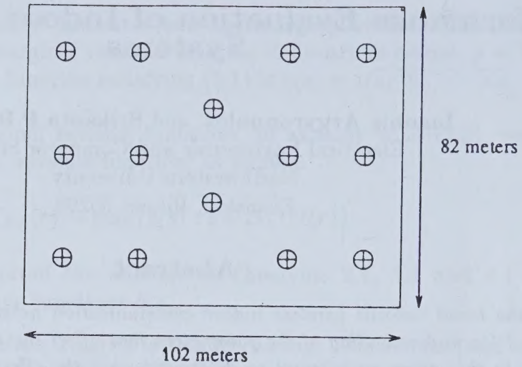


Figure 1: Base Stations per floor

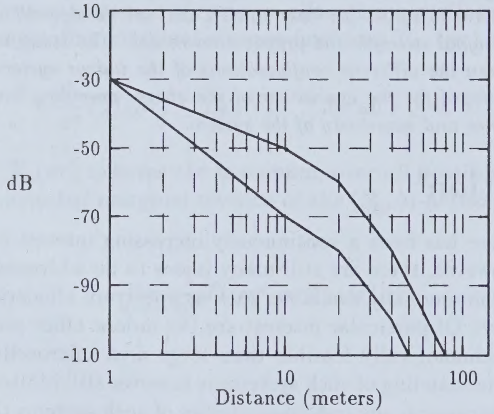


Figure 2: The Distance-Attenuation model

for the choice of this system is its simplicity. This way, the effects of the number of time slots per frequency are simply nonexistent.

2.1 Traffic and Attenuation model

The population of portables in the building is finite. The call origination per portable follows a Poisson process with average 4 calls per hour. The duration of a call is exponentially distributed with mean 100 seconds. Thus, each portable generates 0.111 Erlangs of traffic.

The piecewise linear distance-attenuation model of fig. 2 has been used for the calculation of the signal strength of portables or Base Stations. It has been derived [1] from propagation measurements in a similar indoor environment to the one under study. It is assumed that there is always a line-of-sight between Base Station and any portable that corresponds to an active call. The model consists of two curves corresponding to a maximum and minimum attenuation, and a value is chosen imbetween, according to

Gaussian or Uniform distribution. There is also an extra attenuation of 10 dB for each floor the signal is crossing.

2.2 Operation

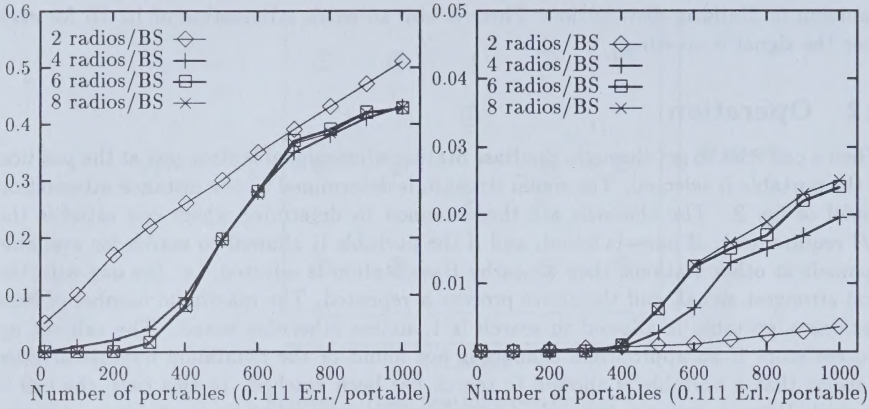
When a call tries to get through, the Base Station whose signal is strongest at the position of the portable is selected. The signal strength is determined by the distance-attenuation model of fig. 2. The channels are then scanned to determine which one satisfies the C/I requirement. If none is found, and if the portable is allowed to search for available channels at other stations, then a nearby Base Station is selected, i.e. the one with the next strongest signal, and the above process is repeated. The maximum number of base stations a portable is allowed to search is 1, unless otherwise noted. The call set up process stops if an appropriate channel is not found or the maximum number of Base Stations that a portable is allowed to search has been reached. In this case, the call is blocked. When a call gets through, a signal level is selected at the portable so that the C/I of the call is above the predefined level of 21 dB.

When a call is established, it is interfering with other calls using the same channel. If the C/I of a call drops below a predefined threshold (21 dB throughout the whole process), a handoff process is initiated. If the portable has the capability of boosting the signal strength level, it tries to improve the signal quality by doing so. In case this fails, the available channels at the current Base Station are searched for one that fulfills the C/I requirement. In case such a channel exists, an *intra-cell* handoff occurs, or simply a channel change. The next step for the portable is to locate the next strongest Base Station, as in the new call set up process. The same restrictions regarding number of Base Stations and channels per Base Station to be searched apply in the case of a handoff. If at any time the C/I drops below 10 dB, then the call is dropped (Forced Termination). If a channel is found, then the call is transferred to the Base Station with the available channel. There is a time restriction between successive handoffs, in order to avoid multiple "back and forth" handoffs from a call that has a poor C/I ratio.

3 Results

The impact of variable number of radios per Base Station is studied first. As shown in fig. 3(a), there is a significant improvement in the call carrying capacity of the system when the number of radios/BS is increased from 2 to 4. The impact of further increase of the number of channels/BS is still positive, but less noticeable. This occurs because the system is interference limited, and increasing the number of radios/BS does not affect the performance of the system significantly beyond a certain point. That is, each Base Station has more channels available, which, however, cannot be re-used. The improvement is more significant at low loads because the statistics are dominated by the probability of finding a free radio. At high loads, the probability of finding an available channel is more important than the probability of finding a free radio.

From fig. 3(b) it is obvious that at low loads, all configurations have comparable blocking probabilities. At high loads however, 2 radios/BS performs better than the other configurations. This is because the lack of available radios prevents calls from being packed too close to each other (in neighboring cells), therefore lowering the amount of interference.



(a) (b)

Figure 3: (a) Blocking probability for 2, 4, 6 and 8 radios/BS. (b) Probability of Forced Termination for 2, 4, 6 and 8 radios/BS.

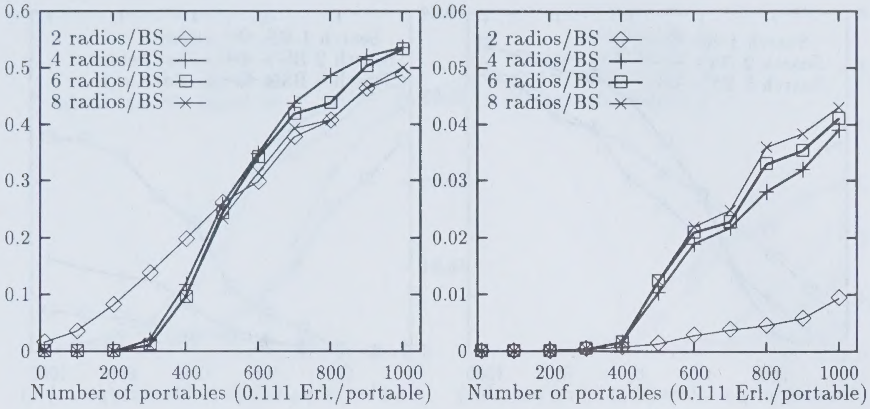
In the next set of results, the effect of varying the number of radios/BS is studied, while also varying the number of Base Stations searched. As shown in fig. 4(a) and 5(a), at low loads, using more radios/BS lowers the blocking probability. As in fig. 3(a), the difference between the 4 curves becomes less as the number of phones in the building is increased. This difference is eliminated as the number of Base Stations searched is increased and the load is heavy. This is because the more Base Stations a portable is allowed to search, the easier it is to find an available radio. Again, as in fig. 3(b), the probability of forced termination in figures 4(b) and 5(b) for 2 radios/BS is lower than all the other configurations due to lack of available channels.

A comparison is done next between searching a different number of base stations while the radios/BS remain constant. In fig. 6(a), at low loads, the probability is dominated by finding an available radio. Thus, the more Base Stations a portable is allowed to search, the lower the blocking probability. At high loads, searching a larger number of Base Stations deteriorates the performance of the system. By allowing the portable to search more Base Stations, it can reach to more distant Base Stations, thus preventing more calls from using the same channel in a greater area. The channel packing is thus reduced and less channels are available to carry calls.

The probability of forced termination is also worse when many Base Stations are searched. In the case where portables are allowed to establish a call with a distant Base Station, the signal quality is poor due to weaker signal, and thus the call is more likely to drop below the acceptable C/I.

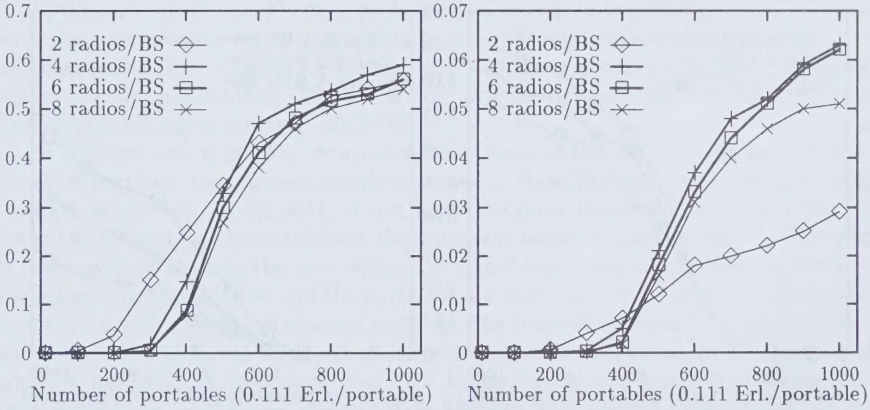
When more channels/BS are allowed, finding a radio is easier, thus the three configurations are comparable at low loads (fig. 7(a)). The statistics are similar to the previous case for high loads, as well as for the probability of Forced Termination.

In the next set of graphs, the nonuniform distribution of the radio fixed parts and the variable signal strength is introduced. Instead of each BS having 4 radios, the BS's closer to the perimeter of the building have 3 radios, while the ones closer to the center, have 5 or 6 radios. The total number of radios for the building remains the same for the



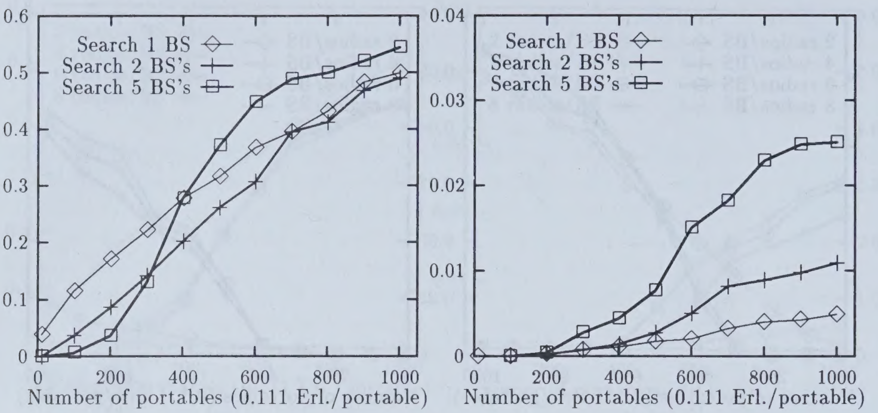
(a) (b)

Figure 4: (a) Blocking probability when 2 Base Stations are searched, for 2, 4, 6 and 8 radios/BS. (b) Probability of Forced Termination when 2 Base Stations are searched, for 2, 4, 6 and 8 radios/BS.



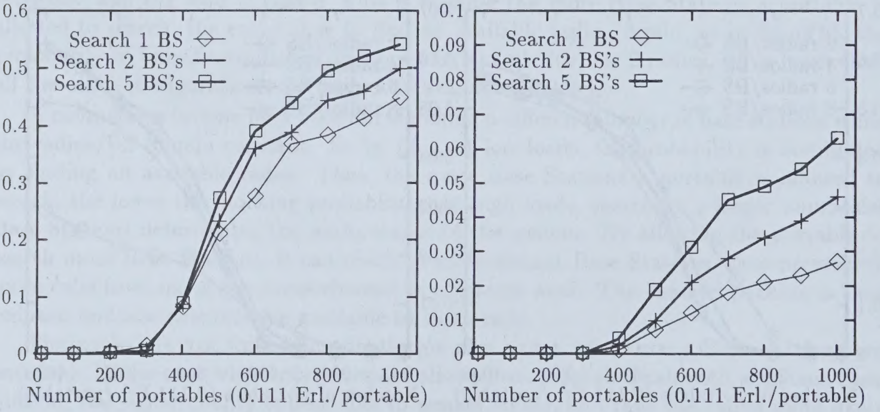
(a) (b)

Figure 5: (a) Blocking probability when 5 Base Stations are searched, for 2, 4, 6 and 8 radios/BS. (b) Probability of Forced Termination when 5 Base Stations are searched, for 2, 4, 6 and 8 radios/BS.



(a) (b)

Figure 6: (a) Blocking probability when 1,2 and 5 Base stations are searched and 2 radios/BS. (b) Probability of Forced Termination when 1,2 and 5 Base Stations are searched and 2 radios/BS.



(a) (b)

Figure 7: (a) Blocking probability when 1,2 and 5 Base stations are searched and 6 radios/BS. (b) Probability of Forced Termination when 1,2 and 5 Base Stations are searched and 6 radios/BS.

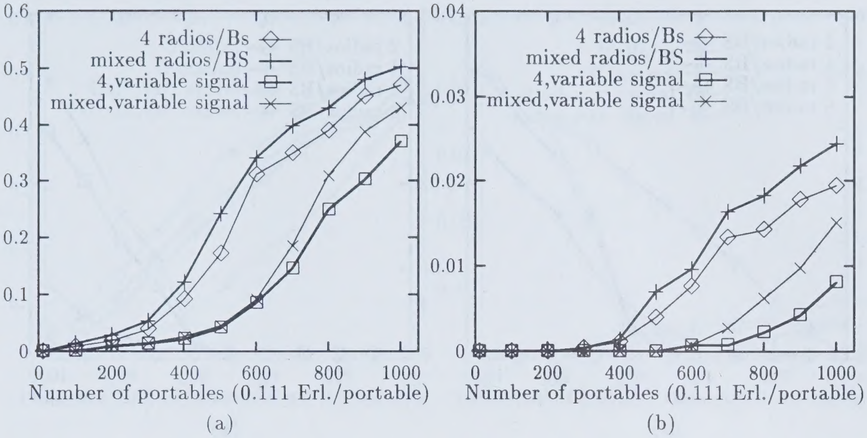


Figure 8: (a) Blocking probability for non-uniform distribution of radios/BS and variable signal strength. (b) Probability of Forced Termination for non-uniform distribution of radios/BS and variable signal strength.

two cases. The idea is that centrally located Base Stations are closer to more locations, thus should have more radios. This is not the case however, as is shown in fig. 8(a). The reason is that even though the BS's close to the center have more radios, they cannot use them efficiently because of the limits imposed by interference.

In the same graph, the blocking probability of the above two configurations is plotted with the portables having the capability to use different levels of signal strength. The blocking probability is lower in this case, and still the non-uniform distribution yields a higher blocking probability. The probability of forced termination is also lower in the case of variable signal strength (fig. 8(b)).

In the next set of graphs, an assessment is made of the effects of using 2, 4, 6 and 8 radios per base station and searching 1 and 2 Base Stations, while variable signal strength is allowed. In fig. 9(a), at low loads the more radios a Base Station has, the lower the blocking probability, since the dominant factor is finding a radio. This effect is more evident than in the case where the signal was constant, because at low loads the interference level is low and the portables are transmitting at low power levels. This results in a higher degree of channel packing. The tradeoff between blocking probability and probability of forced termination still exists, but the difference in the curves of the latter is negligible at low loads. At high loads, having many radios performs worst than having few, in the case where 2 Base Stations were searched (fig. 10(a)). In both cases, after some point the interference "explodes". This is caused because calls are very closely packed and when the interference becomes significant enough to trigger a signal level increase, a chain reaction starts. Every portable is transmitting at maximum power and new calls are prohibited from getting through, thus boosting the blocking probability. In other words, when low signal level is used, the system can support more simultaneous calls, because more channels are being reused. When the power level is increased, the co-channel interference prohibits the channels from being reused.

In the case when 2 Base Stations are searched, the effect of the interference "boom"

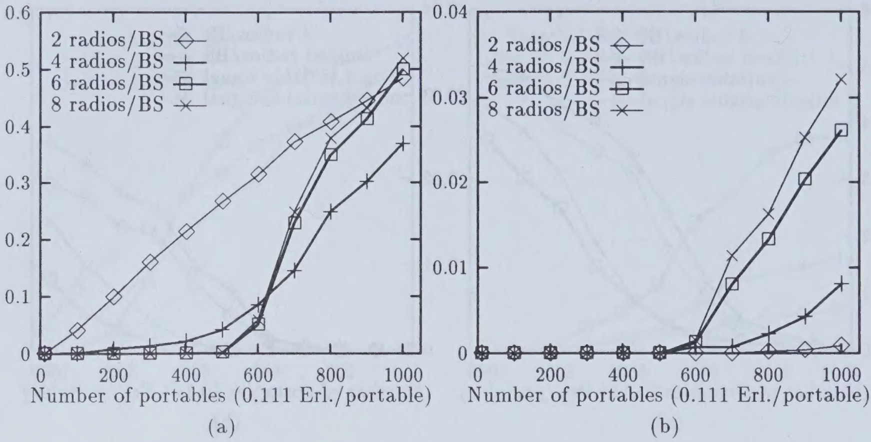


Figure 9: (a) Blocking probability when variable signal strength is used and 1 Base Station is searched (b) probability of Forced Termination when variable signal strength is used and 1 Base Station is searched.

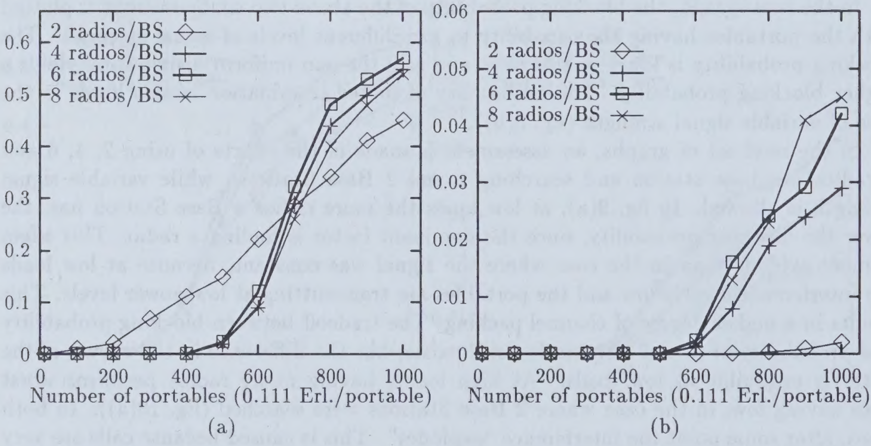
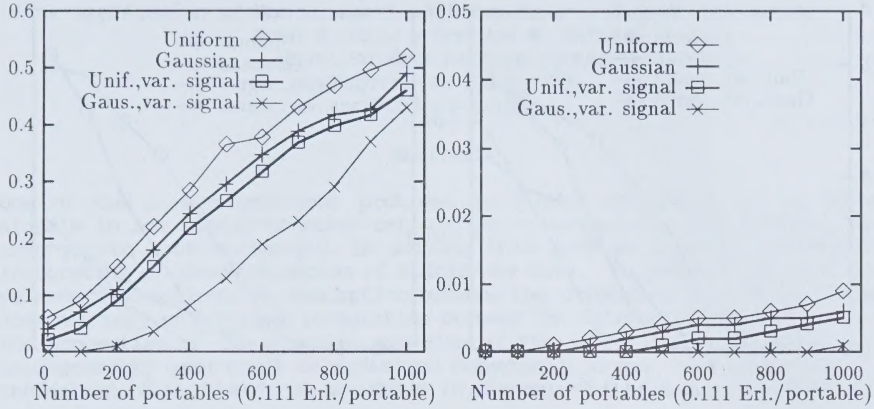


Figure 10: (a) Blocking probability when variable signal strength is used and 2 Base Stations are searched. (b) probability of Forced Termination when variable signal strength is used and 2 Base Stations are searched.



(a)

(b)

Figure 11: (a) Blocking probability for uniform vs. Gaussian distribution and variable vs. constant signal strength for 2 radios/BS. (b) probability of Forced Termination for uniform vs. Gaussian distribution and variable vs. constant signal strength for 2 radios/BS.

is more evident. In this case, when only 2 radios/BS are allowed, the performance is better at high loads. This is because fewer calls are close to each other and thus the interference is maintained at a low level. The above effect is also reflected in fig. 10(b) where all the curves are indistinguishable, until the interference boom occurs.

Next, the effect of Gaussian vs. uniform shadow fading and maximum vs. variable signal strength is investigated, while the number of radios/BS remains constant. In all plots, Gaussian shadow fading performs better than uniform. The reason is that there is less randomness in the signal and portables tend to choose Base Stations that are physically closer to them, which results to better C/I ratio. The Gaussian shadow fading however, does not affect the general behavior of the system. From all the figures (11(a)-12(b)), it is evident that variable signal strength performs better than the maximum and Gaussian shadow fading performs better than uniform. As the number of radios/BS grows, the effect of the interference boom for the variable signal case becomes more evident. In the case of 6 radios/BS the effect is dominant and the variable signal case performs worse than the maximum signal at high loads.

It could be argued that the above assertion is not valid, since in the worst case the portables will be transmitting at maximum power, thus the two curves should be identical. The problem lies in that in the maximum signal power configuration, the channel packing occurs "smoothly", since the interference level is maintained at a constant level. In the variable signal power case however, at some point in time the interference explodes (because all the portables boost their signal level), thus preventing more calls from entering the system.

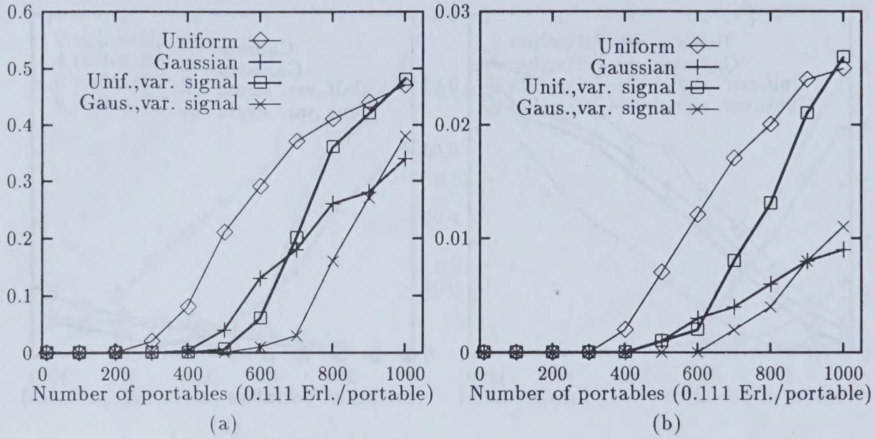


Figure 12: (a) Blocking probability for uniform vs. Gaussian distribution and variable vs. constant signal strength for 6 radios/BS. (b) probability of Forced Termination for uniform vs. Gaussian distribution and variable vs. constant signal strength for 6 radios/BS.

4 Conclusions

In this paper we have studied through simulation the properties of an indoor microcellular system. From the results presented, it is evident that the statistics of the system are dominated by the probability of finding a free radio at low loads and finding a free channel at high loads. Besides, it is not necessary to use more than 4 radios/BS when the traffic is heavy. When the portables are allowed to search more Base Stations, the probability of finding an available radio becomes less significant, but the performance of the system deteriorates, especially at high loads. We have also shown that, at low loads, the performance can be improved by allowing the portables to change the signal strength. Furthermore, the quantitative performance of the system does not seem to be affected by the choice of attenuation model.

References

[1] D. Akerberg, "Properties of a TDMA pico-cellular office communication system", IEEE Proceedings, Vehicular Technolofy Conference, 1989, pp. 186-191.

[2] A.O. Fapojuwo, A. McGirr and Saied Kazeminejad, "A Simulation Study of Speech Traffic Capacity in Digital Cordless Telecommunication Systems", IEEE Transactions on Vehicular Technology, vol. 41, no 1, February 1992, pp. 6-16.

[3] M.W. Evans, "CT2 Common Air Interface", British Telecommunications Engineering, vol. 9, no 45, July 1990.

Application of Non-Linear Transformations to Signal Processing

John E. Gray & Brendon S. McDole
Naval Surface Warfare Center
Dahlgren, VA 22448-5000
703-663-1110

Abstract

One of the more significant problems in signal processing is to detect signals in low signal-to-noise ratio (SNR) environments. The methods that have proven most successful in solving this problem involve non-coherent integration of large ensembles of stationary data. We present a method that relaxes the stationarity assumption between the individual sets of data, that does not rely on coherent integration between the data sets. The method uses the properties of the symbolic dynamics of the logistic map applied to the most commonly used model of a desired detectable signal, the sinc function. The method is shown to work for SNR's in the range of -(10-20) dB for various noise distributions.

Part I. Introduction

One of the more significant problems in signal processing is to detect signals in low signal-to-noise ratio (SNR) environments. The methods that have proven most successful in solving this problem involve non-coherent integration of large ensembles of stationary data [1]. The method presented relaxes the stationarity assumption between the individual sets of data, since it doesn't rely on coherent integration between the data sets. The symbolic dynamics properties of the logistic map are applied to the commonly used model, the sinc function, of a signal. The method is shown to work for SNR's in the range of -(10-20) dB for various noise distributions.

Exponential growth is encountered in many different situations [2]. In biology, the model is population growth represented by a linear population N(t). The difference between birth and death is represented by σ. So the rate of growth is represented by the equation $\frac{dN}{dt} = \sigma N(t)$. If σ is greater than zero, there is exponential growth, while if σ is less than zero the population decreases toward zero. A more realistic model is the logistic equation $\frac{dN}{dt} = \sigma N(t)(1 - \frac{N(t)}{K})$ where K is the carrying capacity. This equation represents saturation of the population. There are two equilibrium points, N(t) = 0, and N(t) = K. A perturbation of an equilibrium is stable if a small change of the location of the point returns to the equilibrium point, it is unstable if it does not. The point is neutral if it remains at the same location. The solution is

$$N(t) = \frac{Ke^{\sigma t}N(0)}{(K + N(0)(e^{\sigma t} - 1))}. \tag{1}$$

This model has proven to be in good agreement with experimental data. The logistic equation is the simplest non-linear difference equation that has interesting characteristics given by placing the logistic equation in discrete form as $N(n+1) = N(n)(a - bN(n))$ which with a change of variable $N(n) = \frac{ax(n)}{b}$ becomes

$$x(n+1)=ax(n)(1 - x(n)). \tag{2}$$

The stable points are solutions to the equation $x^* = f(x^*)$, so for the logistic equation the fixed points are $x^* = 0$ and $x^* = 1-1/a$. A stable point [3] is attractive if for a fixed x^*, a point close to the point, upon iteration is closer to it. Thus if

$$\begin{gathered} x(n+1) = x^* + \varepsilon(n+1) \\ with \left|\frac{\varepsilon(n+1)}{\varepsilon(n)}\right| < 1, \end{gathered} \tag{3}$$

then by a Taylor series expansion

$$x(n+1) = x^* + \varepsilon(n) f'(x^*) \tag{4}$$

which is stable if $|f(x^*)| < 1$. These concepts can be generalized from the behavior of points to functions

Certain functions can be shown to iterate in a manner similar to attractive points for iterative maps. For example, the normalized sinc function has a maximum and zeros which are fixed points of the logistic map, hence they are attractive. The first few iterations of the function are $y_1 = \mu A sinc(\beta x) - \mu A^2 sinc^2(\beta x)$, and $y_2 = A\mu^2 sinc(\beta x) - \mu^2 A sinc^2(\beta x) - \mu^3 A^2 sinc^2(\beta x)(1 - sinc(\beta x))^2$. After sufficient iteration, one term remains fixed while the remainder of the terms tend to oscillate within a narrow band. In a similar fashion the signal, sinc(x) + noise will iterate in a manner that preserves the fixed point for some noise distributions. This makes it possible to detect this type of signal in high noise environment. Figure 1 shows the progression of the iterative process from the initial signal at zero iterations to the final output of the logistic map at 500 iterations, which is the number of iterations used throughout the testing. Although it is quite obvious that the non-linear transform finds the main peak after 25 iterations, later applications require a higher number of iterations to overcome the noise level.

Part II. Discussion of Results

The signal for the sinc function was generated using the following:

$$\vec{\omega} = [\Theta_1, \Theta_2] \tag{5}$$

$$y(\vec{\omega}) = \frac{\sin((\vec{\omega} - \Theta_c)\pi\beta)}{(\vec{\omega} - \Theta_c)\pi} \tag{6}$$

where θ_1, θ_2 are the upper and lower digital frequencies in radians, θ_c corresponds to the frequency at which the central peak appears, and β is a scaling factor, whose effects are not readily apparent since the signal is normalized. This is shown in Figure 2. Throughout the remainder of the testing, β is chosen constant with a value of 10. The simulated sinc signal serves as the test signal for all results shown in Figure 4 through Figure 14. Figures 4 through 6 show the results and interpretations of the application of the logistic map to the signal shown in Figure 3, which is the original signal after being modified by the addition of a uniformly distributed noise vector. Figure 4 shows the resulting logistic map after the transform is applied. The minimum point on this graph corresponds in position to the peak in Figures 2 and 3, which is graphically interpreted in Figures 5 and 6. In all the following tests, μ is fixed at 3.4, the signal to noise ratio (SNR) remains set to a constant -10 dB, and the non-linear transform performs 500 iterations per run. The choice of μ is based on experiment and not with any theoretical justification. A value just before the onset of chaos seems to work best.

The dashed line in Figures 5 and 6 represent the graphical interpretation of Figure 4 with respect to the noise-free and noisy signal. The overlays in Figures 5 and 6 demonstrate the successful determination of the peak's position by application of the transform. Figures 7 through 10 show the results based on the simulated sinc signal shown in Figure 2 after adding a Gaussian distributed noise vector. All other definable parameters for the application of the attractor were held constant. Figure 7 shows the noisy signal; Figure 8 shows the resulting logistic map after the application of the attractor. Figures 9 and 10 are overlays of the clear and noisy signal (solid line) and the graphical interpretation of the logistic map (dashed line). Figures 9 and 10 show the successful determination of the main peak in a signal which when compared to the noisy signal in Figure 3, is considerably more obscure based on the relative noise floors. Figures 11 through 14 show the test signal and results for the last phase of the first test set. In this part of the testing the original sinc signal was modified by adding a Rayleigh distributed noise vector. The resulting signal is shown in Figure 11. Figure 12 shows the resulting logistic map, and Figures 13 and 14 again show an overlay of the graphical interpretation of the logistic map (dashed line) and the clear and noisy signal (solid line). Figures 13 and 14 again demonstrate that the non-linear transform is a successful technique for determining the location of the main spectral peak.

In reviewing the first test set, the results shown indicate that the application of the non-linear transform for peak determination was quite successful for a SNR of -10 dB. When the signals having their noise modeled as either uniform or Rayleigh distributions are compared to the signal with Gaussian distributed noise, it appears that the non-linear transform would be able to successfully extract their peaks at a SNR lower than -10 dB. The first test set demonstrated the success of the application of the non-linear transform in determining the position of a single spectral peak; the second test set determines the ability of the non-linear transform to indicate the presence of multiple peaks. For the second test set, the signal in Figure 15 was generated as the basis for the testing.

The spectrum, $Y(\omega) = |-\frac{\omega}{\omega_0}G(\omega)|$ in Figure 15, has positive component $G_p(\omega)$ and a negative component $G_n(\omega)$ was generated using the component for the negative frequency portion

$$G_n(m) = G_n(0) + 2(-1)^m J_m(z)\, e^{j2nT(\omega_p - m\Omega - \omega)}\, sinc((\omega_p - \omega - m\Omega)T), \tag{7}$$

and the positive frequency portion

$$G_p(m) = G_p(0) + 2J_m(z)\, e^{j2nT(\omega_p + m\Omega - \omega)}\, sinc((\omega_p - \omega + m\Omega)T) \tag{8}$$

which represent a signal with multiple sidebands caused by inner-modulation, where $z = \frac{2\omega_0 A_0}{3\times10^8}$, $\omega_p = (1-2\beta)\omega_0$, $\omega_c = \frac{\omega_0(1-2v_0)}{3\times10^8}$, $\beta = \frac{v_0}{3\times10^8}$, the interval is defined as

$$\begin{gathered} \omega = (\omega_c - 300) + \Delta\omega \\ \omega \in [\omega_c - 300, \omega_c + 300] \end{gathered} \tag{9}$$

and the constants are Ω =50 Hz, A_0 = .05 m, T = .128 sec, v_0 = 100 m/s, and $\omega = 10^{10}$ rad/sec. These parameters were chosen for illustrative purposes only.

The technique used to extract multiple peaks from a given signal was a recursive search-interpret-filter process. The search-interpret process mirrors the process used for single peak detection; the filter process basically removes the detected peak from the signal and normalizes the new signal. The whole process is then repeated. In the following tests, the recursion was limited to four loops; however this limit does not necessarily restrict the number of detected peaks to four. In the cases where multiple peaks have approximately equivalent magnitudes, the attractor oscillates between the two, which is readily apparent from the logistic map. In the following tests, the graphs of the logistic maps are omitted, since they are constantly updated. Figures 16 through 18 show the results for the first phase of the second test set. Figure 16 shows the signal after modification the addition a uniformly distributed noise vector. Figure 17 shows the signal after being modified by a detection process. A new, binary-based vector of the same length as the original signal is created, having values of either 0.1 or 1, indicating the absence or presence of a peak, respectively. The new vector is an image of the graphical interpretation of several logistic maps and serves as a type of mask, which is then multiplied element by element with the original signal.

Figure 18 is an overlay of the original signal and the modified signal seen in Figure 17. The original peak is inverted and removed from the modified peak to allow easier positional comparisons. From this figure it can be seen that the multiple peak detection is effective to a certain degree. The process overlooked one peak in the original signal, and added two 'phantom' peaks, which are indicated on the graph by the asterisks. Figures 19 through 21 constitute the second phase of the second test set. In Figure 19, the signal has been modified by the addition of a Gaussian distributed noise vector. Figure 20 shows the signal after the application of the non-linear transform and the mask, as described above. Figure 21 shows an overlay of the original signal (dashed line) and the modified signal (solid line).

There are only two correct detections in this Figure, with many near misses. The pair of sidebands inside ±10 GHz range in the original signal are mirrored in an offset pair of sidebands that occur in the same region in the modified signal; this phenomenon also occurs for other sidebands in the signal, and appears to be an effect caused by the addition of the noise vector. The results from the last phase of the second test set are shown in Figures 22 through 24. Figure 22 is the original signal plus a Rayleigh distributed noise vector, and Figure 23 shows the signal after the application of the non-linear transform. The offset-mirroring phenomenon appears again in Figure 24, although three correct detections were made. In the cases where the modified signal peaks differ in position from the original peaks, the distance between actual and detected peaks are approximately equivalent on each side of the main lobe; this tends to indicate that the peaks are indeed skewed during the noise addition process, and the non-linear transform correctly identifies the skewed peaks in their transformed positions.

Part III. Conclusion

The results indicate that the application of the non-linear transform $x_{n+1} = \mu x_n (1 - x_n)$ as method of peak detection, specifically tested for signals based on the sinc function and other signals with few distinct peaks, is successful. Part of the reason for the success of this technique is the effect of the transformation on the noise distribution. The resultant PDF of the transformation redistributes the noise in a region, which can be shown for individual PDF's. The effect of this redistribution is to effectively lower the SNR. This result is not a contradiction with the matched filter concept, since the transformation is non-linear which results in a different noise distribution, hence a different effective SNR. This effect will be discussed more fully elsewhere. Additionally an explanation for the effect of the parameter μ needs to be explored. An additional area of research would be to extend this technique to other signals with a different functional dependence. These non-linear transforms would be designed to specifically enhance peaks in low SNR environments, provided the noise is redistributed properly. The transformations effects on PDF's would have to be carefully explored. Testing with real data would provide an additional check of this technique as well as a comparison with other signal enhancement techniques should also be explored. This is a new area with many possible areas to be explored.

References

1. Van Trees, Harry L., Detection, Estimation, and Modulation Theory (John Wiley & Sons, Inc. 1968)

2. May, Robert M., Simple Mathematical Models with Very Complicated Dynamics, Nature, Vol. 261 June 10, 1976, pp. 459-467

3. Li, Tien-Yien, and Yorke, James A., Period Three Implies Chaos, American Mathematical Monthly Vol. 82, No. 10, December 1975, pp. 985-992

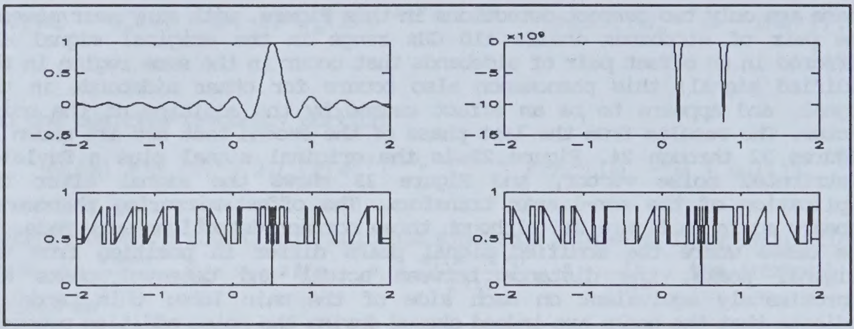


Figure 1

a. sinc function (0 iterations) b. sinc function (5 iterations)
c. sinc function (25 iterations) d. sinc function (500 iterations)

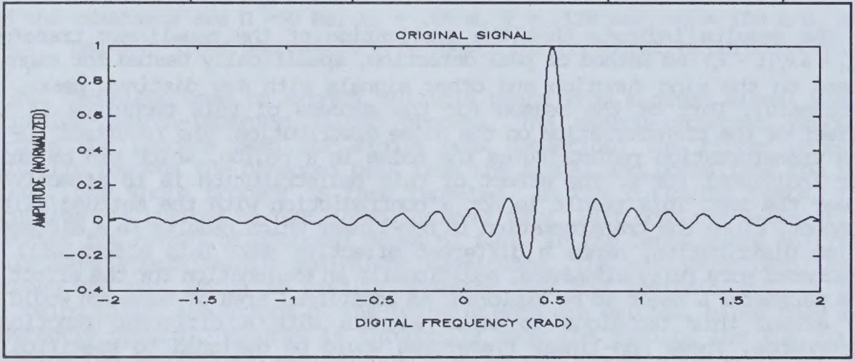


Figure 2

ORIGINAL SIGNAL

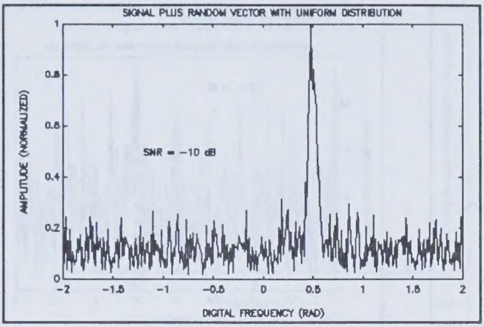


Figure 3

SIGNAL PLUS UNIFORM NOISE

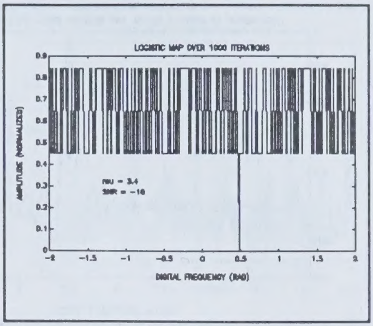


Figure 4

APPLICATION OF LOGISTIC MAP

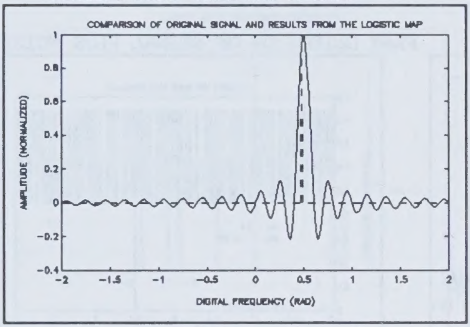


Figure 5

PEAK DETECTION SIGNAL

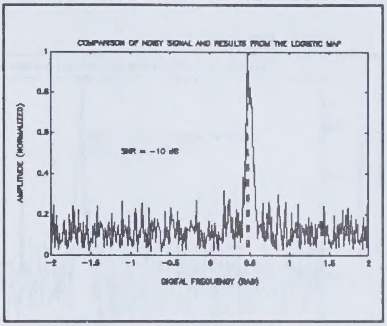


Figure 6

PEAK DETECTION SIGNAL PLUS NOISE

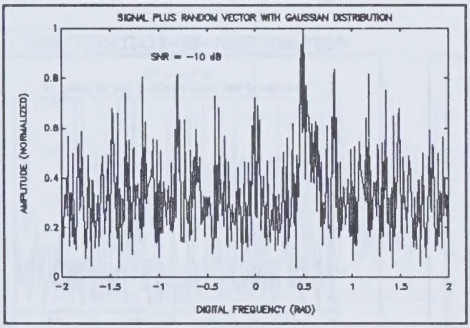


Figure 7

SIGNAL PLUS GAUSSIAN NOISE

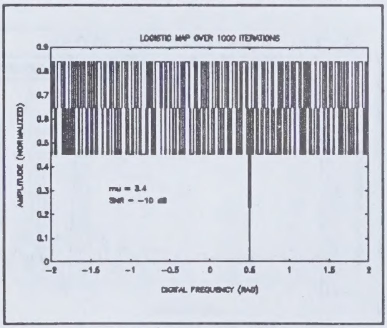


Figure 8

LOGISTIC MAP

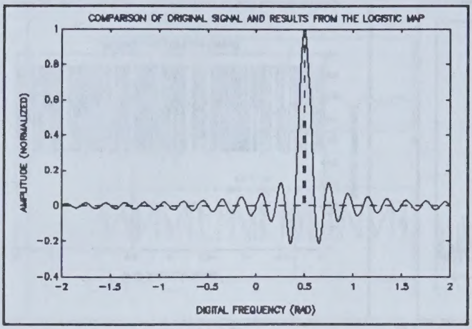


Figure 9

PEAK DETECTION

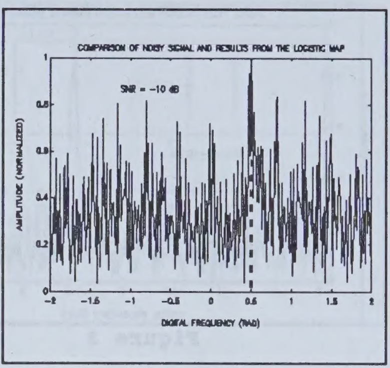


Figure 10

PEAK DETECTION OF SIGNAL PLUS NOISE

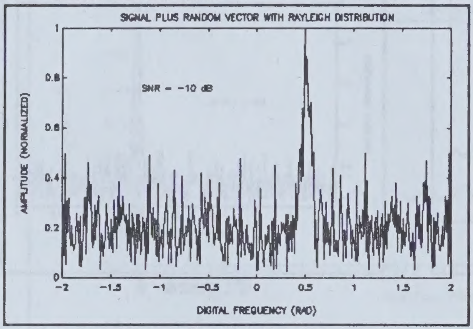


Figure 11

SIGNAL PLUS RAYLEIGH NOISE

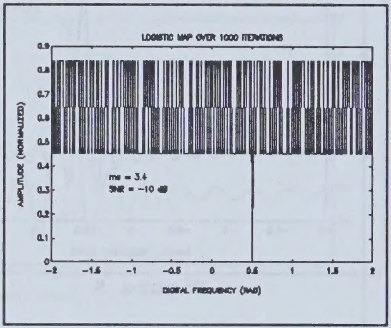


Figure 12

APPLICATION OF LOGISTIC MAP

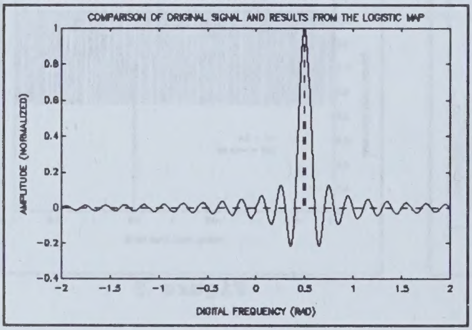


Figure 13

PEAK DETECTION

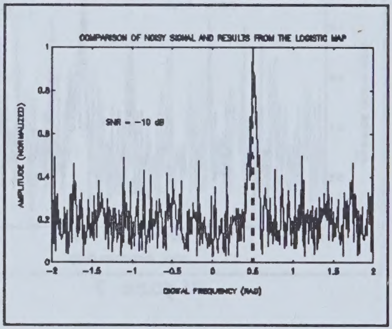


Figure 14

PEAK DETECTION OF SIGNAL PLUS NOISE

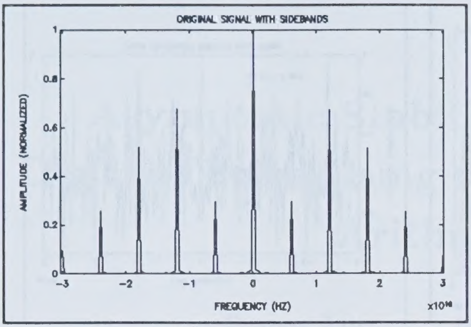


Figure 15

MULTIPLE SIDEBANDS

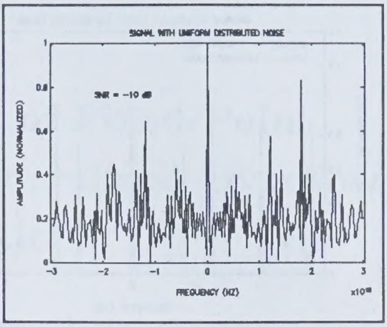


Figure 16

SIGNAL + NOISE

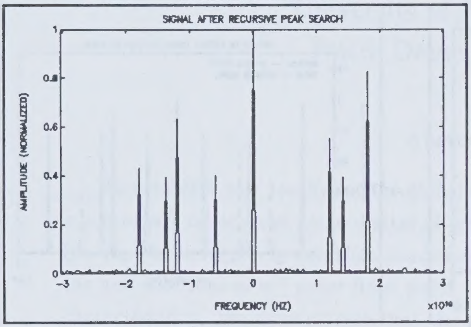


Figure 17

PEAK DETECTION

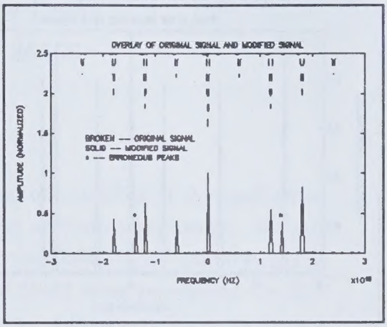


Figure 18

COMPARISON OF RESULTS

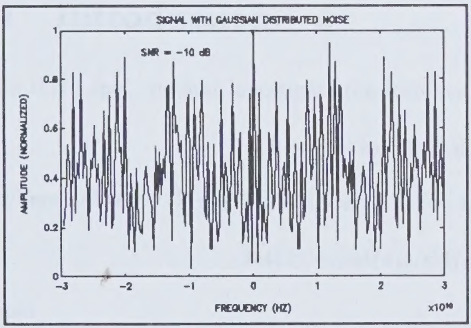


Figure 19

GAUSSIAN NOISE

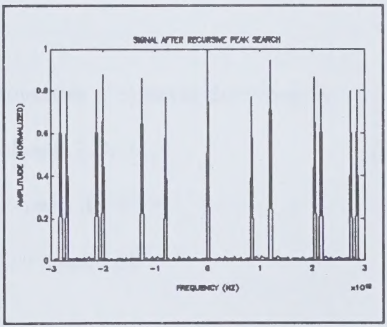


Figure 20

PEAK DETECTION

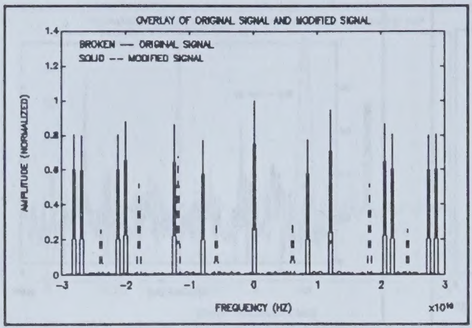


Figure 21

COMPARISON OF TWO METHODS

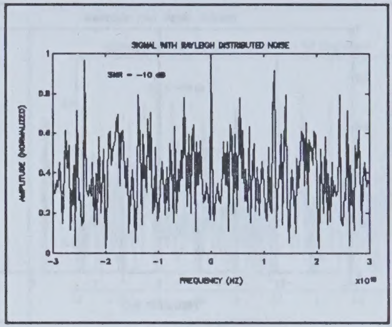


Figure 22

SIGNAL PLUS RAYLEIGH NOISE

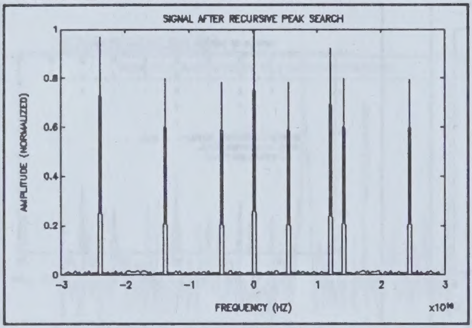


Figure 23

PEAK DETECTION

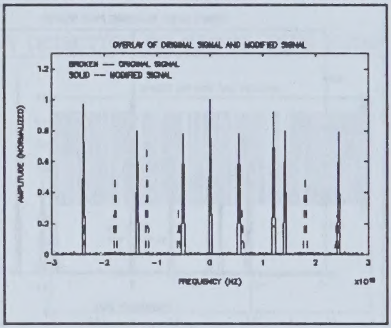


Figure 24

COMPARISON OF RESULTS

Asymptotic Stability of Fixed-Point Digital Filters Using Generalized Overflow Arithmetic *

Derong Liu Anthony N. Michel
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

We establish new results for the global asymptotic stability of the equilibrium $x = 0$ of n^{th} order fixed-point digital filters using saturation arithmetic. We utilize the above results to establish conditions for the global asymptotic stability of the null solutions of n^{th} order fixed point digital filters using generalized overflow characteristics. We demonstrate that the present results are easier to apply and are less conservative than corresponding existing results.

I Introduction

In this paper, we first investigate the stability properties of systems described by

$$x(k+1) = \mathrm{sat}[Ax(k)], \quad k = 0, 1, 2, \cdots. \tag{1}$$

where $x(k) \in D^n \overset{\triangle}{=} \{x \in R^n : -1 \le x_i \le 1,\ i = 1, \cdots, n\},\ A \in R^{n \times n}$,

$$\mathrm{sat}(x) = [\mathrm{sat}(x_1), \mathrm{sat}(x_2), \cdots, \mathrm{sat}(x_n)]^T,$$

and

$$\mathrm{sat}(x_i) = \begin{cases} 1, & x_i > 1 \\ x_i, & -1 \le x_i \le 1 \\ -1, & x_i < -1 \end{cases}.$$

Equation (1) may be employed to represent fixed-point digital filters using saturation overflow arithmetic under zero input. This model does not include quantization effects.

---

*This work was supported by the National Science Foundation under grant ECS 91–07728.

The existence and non-existence of limit cycles in digital filters (under zero input) due to overflow nonlinearities have been investigated extensively (see, e.g., [1]–[9], [11]–[22]). The types of characteristics considered in these studies include zeroing, two's complement, triangular, saturation, and other types of nonlinearities. Since stable *second order* direct form digital filters using saturation arithmetic have been shown to be free of limit cycles (cf. [11], [12]), filters of *any order,* endowed with saturation nonlinearities have received special attention. In addition, as pointed out in [9], [17], [18], conditions obtained for the absence of nonlinear oscillations (under zero input) in digital filters with saturation overflow nonlinearities are generally less conservative than corresponding conditions obtained for digital filters with other types of overflow characteristics.

II Some Existing Results

An early result of Barnes and Fam [1] states that if

$$\|A\|_2 = \sqrt{\lambda_M(A^T A)} < 1 \tag{2}$$

where $\|\cdot\|_2$ denotes the norm of a matrix induced by the l_2 vector norm, A^T represents the transpose of A, and $\lambda_M(A^T A)$ denotes the maximum eigenvalue of $A^T A$, then the digital filter (1) is free of limit cycles. It turns out that this result is true for many other types of overflow nonlinearities. An extension to this result is that the matrix Q in (3) be positive semidefinite, assuming that A is stable, *i.e.*, assuming that every eigenvalue λ_i of A satisfies $|\lambda_i| < 1$,

$$Q = D - A^T D A \geq 0, \tag{3}$$

where D is a diagonal matrix with positive diagonal elements [2]–[4]. This result can not be applied to the case in which the absolute values of some diagonal elements in matrix A are greater than or equal to 1. Another extension to condition (2) is given by

$$\|A\|_p < 1, \quad \text{for some } p, \quad 1 \leq p \leq \infty \tag{4}$$

where $\|\cdot\|_p$ denotes the matrix norm induced by the l_p vector norm [7].

Another time domain result, which has no obvious relations with any of the results cited above, states that if

$$\rho(|A|) < 1, \tag{5}$$

where $\rho(\cdot)$ denotes the spectral radius and $|A| = (|a_{ij}|)$, then the digital filter (1) is free of limit cycles (cf. [5]–[7]). This result is especially useful for testing a digital filter with lower or upper triangular coefficient matrix.

It is shown by Singh [8], [9], that the frequency domain condition

$$D + DA(zI - A)^{-1} + [DA(zI - A)^{-1}]^* \geq 0, \quad \text{for all } |z| = 1, \tag{6}$$

is equivalent to condition (3), where I denotes the $n \times n$ identity matrix, z is a complex variable, and $*$ represents the conjugate transpose. An improvement to condition (6), assuming saturation arithmetic in the digital filters, given by

$$2D + DA(zI - A)^{-1} + [DA(zI - A)^{-1}]^* \geq 0, \quad \text{for all } |z| = 1, \tag{7}$$

is also due to Singh [9]. Note that in (6) and (7), D is still assumed to be a diagonal matrix with positive diagonal elements and that A is assumed to be a stable matrix.

We note that conditions (2)–(7) constitute also conditions for the global asymptotic stability of the null solutions of the digital filters under investigation (with no external inputs).

III Digital Filters Using Saturation Overflow Arithmetic

We will make the following assumption in deriving our results.

Assumption (A–1) Let $x_s = \text{sat}(x) = [\text{sat}(x_1), \cdots, \text{sat}(x_n)]^T$ for $x \in R^n$ and let $H \in R^{n \times n}$ denote a positive definite matrix. Assume that

$$x_s^T H x_s < x^T H x,$$

whenever $x \notin D^n$, $x \in R^n$. ∎

The next result gives a *necessary and sufficient* condition for matrices to satisfy Assumption (A–1). This result is very useful in applications.

Lemma 1 An $n \times n$ positive definite matrix $H = (h_{ij})$ satisfies Assumption (A–1) *if and only if*

$$h_{ii} \geq \sum_{j=1, j \neq i}^{n} |h_{ij}|, \quad i = 1, \cdots, n. \tag{8}$$

The proof of this lemma is omitted because of space limitations.

We are now in a position to establish the following result.

Theorem 1 The equilibrium $x_e = 0$ of (1) is globally asymptotically stable, if A is stable and if there exists a matrix H which satisfies Assumption (A–1), such that $Q \triangleq H - A^T H A$ is positive semidefinite. ∎

Proof: Let us choose $v(x(k)) = x^T(k) H x(k)$ for the system (1). The function v is clearly positive definite and radially unbounded. Also, since

$$Dv_{(1)}(x(k)) = v(x(k+1)) - v(x(k))$$

$$
\begin{aligned}
&= [\mathrm{sat}(Ax(k))]^T H [\mathrm{sat}(Ax(k))] - x^T(k) H x(k) \\
&\leq x^T(k)(A^T H A - H) x(k),
\end{aligned}
$$

and since $H - A^T H A$ is positive semidefinite, $Dv_{(1)}(x)$ is negative semidefinite for all $x(k)$. Therefore, the equilibrium $x_e = 0$ is stable. To show that it is asymptotically stable, we must show that $x(k) \to 0$ as $k \to \infty$ (cf. [10]).

Let us consider an n consecutive step iteration for the system (1), from $n_0 \geq 0$ to $n + n_0$. Without loss of generality, assume that the system (1) saturates at $k = l$, $l \in [n_0,\, n + n_0)$. In view of (A–1), it follows that

$$
\begin{aligned}
v(x(l+1)) &= x^T(l+1) H x(l+1) \\
&= [\mathrm{sat}(Ax(l))]^T H [\mathrm{sat}(Ax(l))] < [Ax(l)]^T H A x(l) \\
&\leq x^T(l) H x(l) = v(x(l)).
\end{aligned}
$$

On the other hand, if no saturation occurs during this period, then, using the fact that if $H - A^T H A$ is positive semidefinite, then $H - (A^T)^n H A^n$ is positive definite when A is stable (cf. [4]), we have

$$
\begin{aligned}
v(x(n+n_0)) &= x^T(n+n_0) H x(n+n_0) \\
&= [A^n x(n_0)]^T H A^n x(n_0) = x^T(n_0)(A^T)^n H A^n x(n_0) \\
&< x^T(n_0) H x(n_0) = v(x(n_0)).
\end{aligned}
$$

Therefore, we can conclude that for the sequence $\{k : k = 1, 2, \cdots\}$, there always exists an infinite subsequence $\{k_j : j = 1, 2, \cdots\}$, such that $Dv_{(1)}(x(k_j))$ is negative for $x(k_j) \neq 0$, and that $v(x(k)) \leq v(x(k_j))$ for all $k \geq k_j$. Since v is a positive definite quadratic form, it follows that $v(x(k_j)) \to 0$ as $j \to \infty$, and therefore $v(x(k)) \to 0$ as $k \to \infty$. This in turn implies that $x(k) \to 0$ as $k \to \infty$. Thus, the equilibrium $x_e = 0$ of (1) is globally asymptotically stable. ∎

Remark 1 Since no limit cycles can exist in a digital filter (with input zero) if its null solution is globally asymptotically stable, Theorem 1 can be used directly as a criterion for testing the nonexistence of overflow limit cycles in n^{th} order digital filters using saturation arithmetic. We note that since in (3), D is assumed to be a diagonal matrix with positive diagonal elements, Theorem 1 constitutes a generalization of condition (3). ∎

Remark 2 The results given in Theorem 1 are in general less conservative than conditions (3), (4) or (6) and appear to have no direct relationships with conditions (5) and (7). However, Theorem 1 is considerably easier to apply than condition (7), since the latter involves matrix inversions. In Section V, we include a specific example which can be analyzed by Theorem 1, but not by any of the previous results given by equations (2)–(7). ∎

IV Digital Filters Using Generalized Overflow Characteristics

In the sequel, we will consider n^{th} order digital filters described by equations of the form

$$x(k+1) = f[Ax(k)], \quad k = 0, 1, 2, \cdots \tag{9}$$

where $x(k) \in R^n$, $A \in R^{n \times n}$, $f(x) = [\varphi(x_1), \varphi(x_2), \cdots, \varphi(x_n)]^T$, and $\varphi : R \to [-1, 1]$ is piecewise continuous. We call (9) a *fixed-point digital filter using general overflow arithmetic.* For such filters, we will make the following assumption.

Assumption (A–2) Let f be defined as in system (9). Assume that $H \in R^{n \times n}$ is a positive definite matrix and that

$$f(x)^T H f(x) < x^T H x, \tag{10}$$

for all $x \in R^n$, $x \notin D^n$. ∎

To establish our next result, Theorem 2, we require the following preliminary results, Lemma 2 and Lemma 3. The proofs of Lemma 2 and Lemma 3 are omitted.

Lemma 2 Assume that f represents the overflow arithmetic for which φ is defined by

$$\varphi(x_i) = \begin{cases} L, & x_i > 1 \\ x_i, & -1 \le x_i \le 1 \\ -L, & x_i < -1 \end{cases}, \tag{11}$$

where $-1 < L \le 1$. Then, an $n \times n$ positive definite matrix $H = (h_{ij})$ satisfies Assumption (A–2) *if and only if*

$$(1 + L) h_{ii} \ge 2 \sum_{j=1, j \ne i}^{n} |h_{ij}|, \quad i = 1, \cdots, n. \tag{12}$$

Lemma 3 For *generalized overflow arithmetic* given by (cf. Figure 1)

$$\begin{cases} L \le \varphi(x_i) \le 1, & x_i > 1 \\ \varphi(x_i) = x_i, & -1 \le x_i \le 1 \\ -1 \le \varphi(x_i) \le -L, & x_i < -1 \end{cases} \tag{13}$$

where $-1 < L \le 1$, (12) is a necessary and sufficient condition for an $n \times n$ positive definite matrix $H = (h_{ij})$ to satisfy Assumption (A–2). ∎

Theorem 2 The equilibrium $x_e = 0$ of n^{th} order digital filter (9), in which φ is given in (11) or (13), is globally asymptotically stable, if A is stable and if there exists a positive definite matrix H which satisfies (12), such that $Q \triangleq H - A^T H A$ is positive semidefinite.

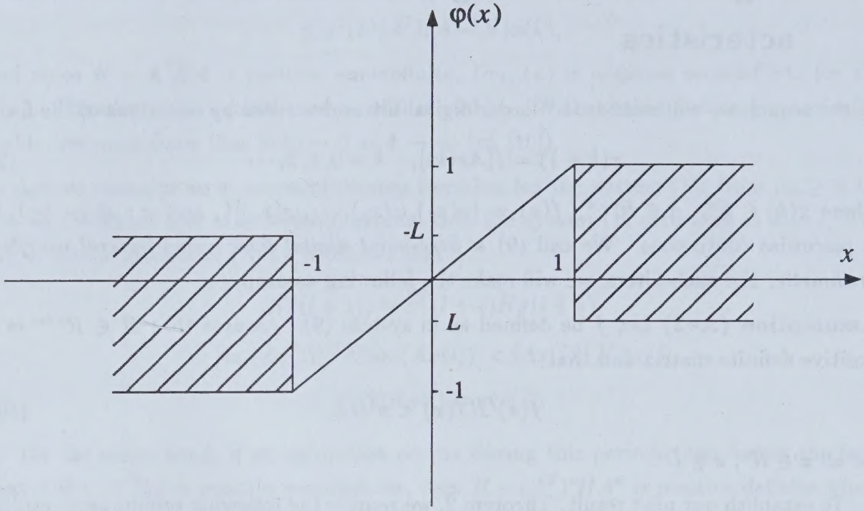


Figure 1: The generalized overflow nonlinearity described by equation (13)

Proof: We can follow the same procedure as in the proof of Theorem 1 to prove that under these conditions, the equilibrium $x_e = 0$ of system (9) is globally asymptotically stable. (Thus the digital filter described by (9) is also free of limit cycles.) ∎

Remark 3 A special case of the overflow characteristics given in (11) is the zeroing characteristic in which $L = 0$. We can also treat the two's complement and the triangular characteristics as special cases of (13) by letting $L \to -1$. In this case, condition (12) will simply mean that matrix H is a *diagonal positive definite matrix.* ∎

Remark 4 For fixed point digital filters employing two's complement or triangular overflow arithmetic, Theorem 2 yields the same result as condition (3), since for these types of arithmetic, the matrix H which satisfies (A–2) must be a *diagonal matrix with positive diagonal elements.* For a digital filter (9) using overflow arithmetic given by (11) or (13), our result in Theorem 2 relaxes the matrix D in (3) from a diagonal matrix with positive diagonal elements to a positive definite matrix H which satisfies the condition (12). This should certainly cover a broader class of stable matrices A. ∎

V Concluding Remarks and Examples

Before considering specific examples, we emphasize that equation (1) describes a large class of discrete time, dynamical systems with saturation nonlinearities (which include important classes of digital filters as special cases).

The results for the global asymptotic stability of the null solutions of system (1) and (9) established in the present paper (Theorem 1 and Theorem 2) are very general, since they involve necessary and sufficient conditions under which a positive definite matrix can be used to generate quadratic form Lyapunov functions for system (1) (Lemma 1, Lemma 2 and Lemma 3). Our results in this paper can be used directly as criteria for testing the nonexistence of overflow limit cycles in n^{th} order digital filters using generalized overflow arithmetic.

The generalized overflow characteristics considered in Section IV cover the usual types of overflow arithmetic used in practice. Our conditions for testing whether a digital filter, using the generalized overflow characteristics, is free of limit cycles, constitute a generalization of condition (3), originally given in [2]–[4]. We generalize the matrix D in (3) from a diagonal matrix with positive diagonal elements to a positive definite matrix satisfying condition (12).

We now consider several specific examples.

Example 1 For the digital filter (1) considered in [9] with A given by

$$A = \begin{pmatrix} 1 & 2^{-3} \\ -0.1 & 0.9 \end{pmatrix}, \tag{14}$$

we have $\|A\|_p > 1,\ p = 1, 2,$ or ∞ , and $\rho(|A|) > 1$. Furthermore, for this filter it can be verified that there is no diagonal matrix D with positive diagonal elements such that $D - A^TDA$ is positive semidefinite. Therefore, conditions (3)–(6) fail as global asymptotic stability tests for this example.

Hypothesis (A–1) is satisfied for this example by choosing

$$H = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 0.8 \end{pmatrix}. \tag{15}$$

Since

$$Q = H - A^THA = \begin{pmatrix} 0.092 & 0.00325 \\ 0.00325 & 0.023875 \end{pmatrix}$$

is positive definite, all conditions of Theorem 1 are satisfied and the equilibrium $x_e = 0$ of system (1) with A specified by (14) is globally asymptotically stable. Therefore, this digital filter is free of limit cycles.

Condition (7) can also be used to ascertain that $x_e = 0$ of the present digital filter is globally asymptotically stable. However, application of condition (7) is extremely involved and cumbersome. ■

Example 2 For the digital filter (1) with

$$A = \begin{pmatrix} 0.6 & -0.2 \\ 0.3 & 1.1 \end{pmatrix}, \tag{16}$$

it can be verified that conditions (3)–(7) are not satisfied.

Choosing H as in (15), it is easily verified that all conditions of Theorem 1 are satisfied. Therefore, the equilibrium $x_e = 0$ of system (1) with A specified by (16) is globally asymptotically stable. Hence, this digital filter is free of limit cycles. ∎

It is extremely difficult to apply condition (7) when the order of the system (1) is greater than 2, as in the next example, where $n = 4$. In particular, the inversion of high-order matrices of variables poses formidable obstacles. On the other hand, the application of Theorem 1 to high-order systems is not particularly difficult.

Example 3 For the system (1) with A given by

$$A = \begin{pmatrix} -1 & 0 & 0.1 & 0 \\ 0.2 & -0.6 & 0 & 0.8 \\ -0.1 & 0.1 & 0.8 & 0 \\ 0.1 & 0 & 0.1 & -0.5 \end{pmatrix}, \tag{17}$$

it can easily be verified that $\|A\|_p > 1,\ p = 1, 2, \text{ or } \infty$, that $\rho(|A|) > 1$, and that there is no diagonal matrix D with positive diagonal elements such that $D - A^T DA$ is positive semidefinite. Hence, conditions (3)–(6) fail as global asymptotic stability tests for the present example (cf. [1]–[8]).

Hypothesis (A–1) is satisfied for this example by choosing

$$H = \begin{pmatrix} 1.4 & 0 & -0.2 & 0.4 \\ 0 & 1.6 & 0.2 & -0.4 \\ -0.2 & 0.2 & 3.4 & 0.5 \\ 0.4 & -0.4 & 0.5 & 3 \end{pmatrix}. \tag{18}$$

Since

$$Q = H - A^T HA = \begin{pmatrix} 0.026 & 0.161 & -0.003 & 0.077 \\ 0.161 & 1.014 & -0.003 & 0.497 \\ -0.003 & -0.003 & 1.124 & 0.774 \\ 0.077 & 0.497 & 0.774 & 0.906 \end{pmatrix}$$

is positive definite, all conditions of Theorem 1 are satisfied, and the equilibrium $x_e = 0$ of the system (1) with such a coefficient matrix is globally asymptotically stable.

From Theorem 2, we see that a $4th$ order digital filter described by (9) with A given in (17), when f represents the zeroing arithmetic, is free of limit cycles since H in (18) satisfies

$$h_{ii} \geq 2 \sum_{j=1, j \neq i}^{n} |h_{ij}|, \quad i = 1, \cdots, n.$$

Indeed, it is also free of limit cycles when generalized overflow arithmetic specified by (11) or (13) is used with $-0.1333 \leq L \leq 1$. ∎

References

[1] C. W. Barnes, A. T. Fam, "Minimum Norm Recursive Digital Filters That Are Free of Overflow Limit Cycles," *IEEE Trans. on Circuits and Systems,* Vol. CAS-24, pp.569–574, Oct. 1977

[2] W. L. Mills, C. T. Mullis, R. A. Roberts, "Digital Filter Realizations without Overflow Oscillations," *IEEE Trans. on A. S. S. P.,* Vol. ASSP-26, pp.334–338, Aug. 1978

[3] M. Kawamata, T. Higuchi, "On the Absence of Limit Cycles in a Class of State-Space Digital Filters Which Contains Minimum Noise Realizations," *IEEE Trans. on A. S. S. P.,* Vol. ASSP-32, pp.928–930, Aug. 1984

[4] P. P. Vaidyanathan, V. Liu, "An Improved Sufficient Condition for Absence of Limit Cycles in Digital Filters," *IEEE Trans. on Circuits and Systems,"* Vol. CAS-34, pp.319–322, March 1987

[5] P. H. Bauer, E. I. Jury, "Stability Analysis of Multidimensional (m-D) Direct Realization Digital Filters Under the Influence of Nonlinearities," *IEEE Trans. on A. S. S. P.,* Vol. 36, pp.1770–1780, Nov. 1988

[6] P. H. Bauer, E. I. Jury, "A Stability Analysis of Two-Dimensional Nonlinear Digital State-Space Filters," *IEEE Trans. on A. S. S. P.,* Vol. 38, pp.1578–1586, Sept. 1990

[7] T. Bose, Mei-Qin Chen, "Overflow Oscillations in State-Space Digital Filters," *IEEE Trans. on Circuits and Systems,* Vol. 38, pp.807–810, July 1991

[8] V. Singh, "Realization of Two's Complement Overflow Limit Cycle Free State-Space Digital Filters: a Frequency-Domain Viewpoint," *IEEE Trans. on Circuits and Systems,* Vol. CAS-33, pp.1042–1044, Oct. 1986

[9] V. Singh, "Elimination of Overflow Oscillations in Fixed-Point State-Space Digital Filters Using Saturation Arithmetic," *IEEE Trans. on Circuits and Systems,* Vol. 37, pp.814–818, June 1990

[10] R. K. Miller, A. N. Michel, *Ordinary Differential Equations,* New York: Academic, 1982

[11] I. W. Sandberg, "A Theorem Concerning Limit Cycles in Digital Filters," *Proc. of the 7th Annual Allerton Conf. on Circuit and System Theory,* Univ. of Illinois, Urbana, pp.63–68, Oct. 1969

[12] P. M. Ebert, J. E. Mazo, M. G. Taylor, "Overflow Oscillations in Digital Filters," *The Bell System Technical Journal,* Vol. 48, pp.2999–3020, Nov. 1969

[13] A. N. Willson, Jr., "Limit Cycles due to Adder Overflow in Digital Filters," *IEEE Trans. on Circuit Theory,* Vol. CT-19, pp.342–346, July 1972

[14] T. A. C. Claasen, W. F. G. Mecklenbräuker, J. B. H. Peek, "Frequency Domain Criteria for the Absence of Zero-Input Limit Cycles in Nonlinear Discrete-Time Systems, with Applications to Digital Filters," *IEEE Trans. on Circuits and Systems,* Vol. CAS-22, pp.232–239, March 1975

[15] T. A. C. Claasen, L. O. G. Kristiansson, "Necessary and Sufficient Conditions for the Absence of Overflow Phenomena in a Second-Order Recursive Digital Filter," *IEEE Trans. on A. S. S. P.,* Vol. ASSP-23, pp.509–515, Dec. 1975

[16] D. Mitra, "Large Amplitude, Self-Sustained Oscillations in Difference Equations which Describe Digital Filter Sections Using Saturation Arithmetic," *IEEE Trans. on A. S. S. P.,* Vol. ASSP-25, pp.134–143, Apr. 1977

[17] I. W. Sandberg, "The Zero-Input Response of Digital Filters Using Saturation Arithmetic,"*IEEE Trans. on Circuits and Systems,* Vol. CAS-26, pp.911–915, Nov. 1979

[18] D. Mitra, "Criteria for Determining if a High-Order Digital Filter Using Saturation Arithmetic is Free of Overflow Oscillations," *The Bell System Technical Journal,* Vol. 56, pp.1679–1699, Nov. 1977

[19] A. G. Bolton, "A Two's Complement Overflow Limit Cycle Free Digital Filter Structure," *IEEE Trans. on Circuits and Systems,* Vol. CAS-31, pp.1045–1046, Dec. 1984

[20] K. T. Erikson, A. N. Michel, "Stability Analysis of Fixed-Point Digital Filters Using Computer Generated Lyapunov Functions–Part I: Direct Form and Coupled Form Filters," *IEEE Trans. on Circuits and Systems,* Vol. CAS-32, pp.113–132, Feb. 1985

[21] V. Singh, "A New Realizability Condition for Limit Cycle-Free State-Space Digital Filters Employing Saturation Arithmetic," *IEEE Trans. on Circuits and Systems,* Vol. CAS-32, pp.1070–1071, Oct. 1985

[22] J. H. F. Ritzerfeld, "A Condition for the Overflow Stability of Second-Order Digital Filters that is Satisfied by all Scaled State-Space Structures Using Saturation,"*IEEE Trans. on Circuits and Systems,* Vol. CAS-36, pp.1049–1057, Aug. 1989

Recursive Direction Finding via RLS

Joseph H. Clements III and James Ting-Ho Lo*
Department of Mathematics & Statistics
University of Maryland Baltimore County
Baltimore, Maryland 21228

Abstract

We have developed an algorithm which recursively updates the maximum likelihood (ML) estimates of arrival directions of multiple signals after each snapshot is taken by a sensor array. This approach is an extension of the method of cyclic regression but it utilizes the recursive least squares (RLS) algorithm to update the steering vector estimates after each snapshot is taken by the sensor array. Numerical results have been included which demonstrate the effectiveness of this new approach to direction finding.

1. Introduction

Batch processing algorithms, such as MUSIC by Schmidt [8], its variants, and various maximum likelihood (ML) estimation techniques [1, 10, 11], have been commonly used to solve the high-resolution direction finding problem. In these methods, multiple signal direction of arrival (DOA) estimates are obtained only after the sample covariance matrix is computed and then processed from a batch of data which was previously collected by the sensor array. Presenting these algorithms with larger batches of data generally results in better estimates although the amount of data which must be stored and the amount of numerical computations which must be performed must also be increased.

A major deficiency with using these types of algorithms as the solution to the direction finding problem is in the handling of additional data when it becomes available from the sensor array. The sample covariance matrix can easily be updated to include the new data, however the DOA estimates must be completely recomputed. Consequently, in situations where data is constantly being collected and optimal estimates are continually required, these batch processing algorithms are computationally inefficient.

Thus there is a need for a recursive solution to the direction finding problem. We will present one such solution. We have derived an algorithm which updates the DOA estimates after each snapshot is taken by the sensor array. We require only that the previous estimates of the DOAs and the estimated steering matrix and its covariance be stored prior to the processing of the new data. Therefore, once the updated estimates are obtained, the new data can be deleted. Moreover, the estimates which are produced using this recursive algorithm converge to the ones which are obtained using the less efficient batch processing algorithms.

Our approach is based on the method of cyclic regression (CR) which was developed by Lo [4, 5, 6, 7] for efficient direction finding and sensor localization and calibration. We will only address the problem of multiple signal direction finding in this paper. However,

---

*This work was supported in part by the Office of Naval Research under Grant N00014-91-J-1071.

the derivation of a recursive sensor localization and calibration procedure, which follows most readily from this paper and [6], will be presented at a later date.

Our method, like the CR method, exploits the linearity and separability structure of the three groups of unknown parameters namely, the signal waveforms, the components of the steering vectors, and the signal DOAs. Once separated, the log-likelihood function is then maximized by cyclically varying one set of these unknown parameters at a time while holding the other two fixed. As a result of the linear structure, each step in the cycle is solved using linear regression techniques and thus the highly nonlinear array signal processing problem is decomposed into a finite sequence of linear regression problems. It is this decomposition which allows us to easily keep track of the second order statistics of the estimates of the rows of the steering matrix, thus enabling us to use the RLS procedure to recursively updated their estimates and covariances.

The organization of this paper will now be described. In section 2, we will state the recursive multiple signal direction finding problem, and log-likelihood function for the unknown parameters will be formulated in section 3. We will describe how to estimate the signal waveforms in section 4, and explain how to update the steering vector estimates using the RLS technique in section 5. Then, in section 6, we will detail how the results in sections 4 and 5 can be used to recursively update the signal DOAs. The complete recursive direction finding algorithm will then be summarized in section 7. We will end our presentation with simulation results which will be given in section 8.

2. Problem Statement

Suppose that K narrow-band signals with central frequency ω impinge an array consisting of M coplanar sensors from the directions $\theta_1, \theta_2, \ldots, \theta_K$. If we assume that the sensors and the sources of the signals lie on the same plane, we can construct a Cartesian coordinate system on the plane in which the M sensors which are located at $(x_m, y_m), m = 1, 2, \ldots, M$.

The measurements observed by the sensor array at each time instant t can be described by the following equation

$$\mathbf{z}(t) = \sum_{k=1}^{K} \mathbf{a}(\theta_k) s_k(t) + \mathbf{e}(t), \tag{1}$$

where

$$\begin{aligned}
\mathbf{z}(t) &\stackrel{\text{def}}{=} [z_1(t), z_2(t), \ldots, z_M(t)]^T, & (2)\\
\mathbf{a}(\theta_k) &\stackrel{\text{def}}{=} [r_1 \exp(-j\psi_1(\theta_k)), r_2 \exp(-j\psi_2(\theta_k)), \ldots, r_M \exp(-j\psi_M(\theta_k))]^T, & (3)\\
\psi_m(\theta_k) &\stackrel{\text{def}}{=} \frac{\omega}{c} d_m(\theta_k) + \phi_m, & (4)\\
d_m(\theta_k) &\stackrel{\text{def}}{=} x_m \cos\theta_k + y_m \sin\theta_k, & (5)\\
s_k(t) &\stackrel{\text{def}}{=} \mu_k \exp[j(\omega t + \nu_k)], & (6)\\
\mathbf{e}(t) &\stackrel{\text{def}}{=} [e_1(t), e_2(t) \ldots, e_M(t)]^T, & (7)
\end{aligned}$$

These variables can be described as follows

$\mathbf{z}(t)$ = a "snapshot" taken by the sensor array at time t,
$\mathbf{a}(\theta_k)$ = the "steering vector" of the sensor array towards the k^{th} source,
$\psi_m(\theta_k)$ = the "phase delay" from the origin to m^{th} sensor in the DOA of k^{th} signal,

$$\begin{aligned}
r_m &= \text{the “gain” of } m^{\text{th}} \text{ sensor relative to the origin,}\\
\phi_m &= \text{the “intrinsic phase” of } m^{\text{th}} \text{ sensor,}\\
d_m(\theta_k) &= \text{the distance from the origin to the } m^{\text{th}} \text{ sensor in the DOA of signal } k,\\
s_k(t) &= \text{the } k^{\text{th}} \text{ narrow-band signal centered at frequency } \omega \text{ at time } t,\\
\mu_k &= \text{random amplitude of the } k^{\text{th}} \text{ signal,}\\
\nu_k &= \text{random phase of the } k^{\text{th}} \text{ signal,}\\
\mathbf{e}(t) &= \text{the measurement noise observed by the sensors at time } t.
\end{aligned}$$

If we define the DOA vector θ, the steering matrix $\mathbf{A}(\theta)$ and the signal vector $\mathbf{s}(t)$ as

$$\theta \overset{\text{def}}{=} [\theta_1, \theta_2, \ldots, \theta_K]^T, \tag{8}$$

$$\mathbf{A}(\theta) \overset{\text{def}}{=} [\mathbf{a}(\theta_1), \mathbf{a}(\theta_2) \ldots, \mathbf{a}(\theta_K)], \tag{9}$$

$$\mathbf{s}(t) \overset{\text{def}}{=} [s_1(t), s_2(t) \ldots, s_K(t)]^T, \tag{10}$$

then we can rewrite equation (1) in the following matrix form

$$\mathbf{z}(t) = \mathbf{A}(\theta)\mathbf{s}(t) + \mathbf{e}(t). \tag{11}$$

Problem Statement: Assume that at time t we have the ML estimate $\hat{\theta}(t-1)$ of the DOAs θ and an a priori estimate $\hat{\mathbf{A}}(t-1)$, whose rows have error covariances $\hat{\mathbf{\Lambda}}_m(t-1)$, $m = 1, 2, \ldots, M$, of the steering matrix $\mathbf{A}(\theta)$. Suppose that at time t the sensor array takes a snapshot $\mathbf{z}(t)$. We want to derive a computationally efficient algorithm which computes the ML estimate $\hat{\theta}(t)$ of θ using only $\mathbf{z}(t)$, $\hat{\theta}(t-1)$, $\hat{\mathbf{A}}(t-1)$ and $\hat{\mathbf{\Lambda}}_m(t-1)$, $m = 1, 2, \ldots, M$.

3. Maximum Likelihood Estimation

In this section we will derive the ML equation for estimating the the unknown DOA vector θ and the unknown signal waveforms $\mathbf{s}(t)$ given $\mathbf{z}(t)$, $\hat{\mathbf{A}}(t-1)$ and $\hat{\mathbf{\Lambda}}_m(t-1)$, $m = 1, 2, \ldots M$. We will assume that the rows $\hat{\mathbf{a}}^m(t-1)$, $m = 1, 2, \ldots M$, of $\hat{\mathbf{A}}(t-1)$ are unbiased and uncorrelated. We will also assume that each row $\hat{\mathbf{a}}^m(t)$ can be reliably approximated by a complex gaussian distribution. Therefore, the probability distributions for the rows $\hat{\mathbf{a}}^m(t-1)$ of $\hat{\mathbf{A}}(t-1)$ can be described by the following equations

$$\begin{aligned}
f(\hat{\mathbf{a}}^m(t-1)) &= \pi^{-K} |\hat{\mathbf{\Lambda}}_m(t-1)|^{-1}\\
&\quad \cdot \exp\left\{ - (\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta))\, \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\, (\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta))^H \right\},\\
&\quad m = 1, 2, \ldots, M,
\end{aligned} \tag{12}$$

where $|\cdot|$ denotes the determinant and the superscript H denotes the Hermitian transposition.

In addition, we will assume that the measurement noise sequence $\{\mathbf{e}(t)\}$ is a complex, zero mean, white gaussian sequence with covariance given by $\sigma^2\mathbf{I}_M$. This assumption implies that each snapshot $\mathbf{z}(t)$ is a M-dimensional complex gaussian vector with a probability distribution given by

$$g(\mathbf{z}(t)) = \pi^{-M} \mid \sigma^2\mathbf{I}_M \mid^{-1} \exp\left\{ - \sigma^{-2} \|\mathbf{z}(t) - \mathbf{A}(\theta)\mathbf{s}(t)\|^2 \right\}. \tag{13}$$

In order to simplify our presentation, we will assume that σ^2 is known.

The likelihood function $l(\theta, \mathbf{s}(t))$ of $(\theta, \mathbf{s}(t))$ given the data $\mathbf{z}(t)$ and the a priori estimate $\hat{\mathbf{A}}(t-1)$ is therefore given by

$$l(\theta, \mathbf{s}(t)) = g(\mathbf{z}(t)) \prod_{m=1}^{M} f(\hat{\mathbf{a}}^m(t-1)) . \tag{14}$$

By substituting (12) and (13) into (14) and then taking the logarithm of both sides yields the following expression for the negative of the log-likelihood function $-L(\theta, \mathbf{s}(t))$

$$\begin{aligned} -L(\theta, \mathbf{s}(t)) &= \sigma^{-2}\|\mathbf{z}(t) - \mathbf{A}(\theta)\mathbf{s}(t)\|^2 \\ &+ \sum_{m=1}^{M} \left\{ (\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta)) \hat{\mathbf{\Lambda}}_m^{-1}(t-1) (\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta))^H \right\}, \end{aligned} \tag{15}$$

where the constant terms have been omitted. The ML estimate of $(\theta, \mathbf{s}(t))$ can now be obtained by minimizing this equation with respect to the unknown parameters. This can be denoted by

$$\left(\hat{\theta}(t), \hat{\mathbf{s}}(t)\right) = \arg \min_{(\theta, \mathbf{s}(t))} \left\{ -L(\theta, \mathbf{s}(t)) \right\}, \tag{16}$$

where $\hat{\theta}(t)$ and $\hat{\mathbf{s}}(t)$ are the ML estimates of θ and $\mathbf{s}(t)$, respectively.

In order to determine the solution to this equation, we will first make the following observations:

1. While holding the signal DOAs θ fixed, the signal waveforms $\mathbf{s}(t)$ enter the sensor measurements $\mathbf{z}(t)$ linearly.
2. While holding the signal waveforms $\mathbf{s}(t)$ fixed, the rows $\mathbf{a}^m(\theta)$ of the steering matrix $\mathbf{A}(\theta)$ enter the sensor measurements $\mathbf{z}(t)$ linearly.
3. If we take a Taylor's series expansion of each component of the steering vectors $\mathbf{a}(\theta_k)$ around the best available estimate of the signal DOAs θ and drop the second and higher order terms, the signal DOAs themselves enter the sensor measurements $\mathbf{z}(t)$ linearly.

These observations suggest that the negative of the log-likelihood function can be minimized by cyclically varying one set of these unknown parameters while holding the other two fixed. As a result of the linear structure which was observed, each of these steps can be solved using linear regression techniques.

4. Estimating the Signal Waveforms

In this section we will determine the formula for obtaining the ML estimate of the signal waveforms $\mathbf{s}(t)$. We will assume that the signal DOAs θ are held fixed in which case the maximizer of the log-likelihood function $L(\theta, \mathbf{s}(t))$ is actually the minimizer of the following equation

$$-L(\mathbf{s}(t)) = \sigma^{-2} \| \mathbf{z}(t) - \mathbf{A}(\theta)\mathbf{s}(t) \|^2 . \tag{17}$$

It is well known that the minimizer of this equation with respect to $\mathbf{s}(t)$ is given by the following formula,

$$\hat{\mathbf{s}}(t) = \mathbf{A}^{+}(\theta)\mathbf{z}(t), \tag{18}$$

where $\mathbf{A}^{+}(\theta)$ denotes the pseudo-inverse of $\mathbf{A}(\theta)$. If we assume that the K steering vectors of $\mathbf{A}(\theta)$ are linearly independent, then $\mathbf{A}^{+}(\theta)$ is given by

$$\mathbf{A}^{+}(\theta) \stackrel{\text{def}}{=} \left(\mathbf{A}^{H}(\theta)\mathbf{A}(\theta)\right)^{-1} \mathbf{A}^{H}(\theta). \tag{19}$$

5. Estimating the Steering Vectors

In this section, we will describe how to update the estimated steering matrix $\hat{\mathbf{A}}(t)$ given the new data $\mathbf{z}(t)$ and the a priori estimates $\hat{\mathbf{a}}^m(t-1)$ and covariances $\hat{\mathbf{\Lambda}}_m(t-1)$, $m = 1,2,\ldots,M$. We will assume that the signal waveforms $\mathbf{s}(t)$ are known and held fixed in which case the log-likelihood function $L(\theta, \mathbf{s}(t))$ can be denoted by $L(\theta)$. Since the first term on the right side of (15) can be written as

$$\|\mathbf{z}(t) - \mathbf{A}(\theta)\mathbf{s}(t)\|^2 = \sum_{m=1}^{M} |\mathrm{z}_m(t) - \mathbf{a}^m(\theta)\mathbf{s}(t)|^2, \tag{20}$$

we can rewrite equation (15) as follows

$$\begin{aligned}
-L(\theta) = {} & \sum_{m=1}^{M} \Big[\sigma^{-2}|\mathrm{z}_m(t) - \mathbf{a}^m(\theta)\mathbf{s}(t)|^2 \\
& + (\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta))\,\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\,(\hat{\mathbf{a}}^m(t-1) - \mathbf{a}^m(\theta))^H\Big] \\
= {} & \sum_{m=1}^{M} \Big[\sigma^{-2}|\mathrm{z}_m(t)|^2 - \sigma^{-2}\mathrm{z}_m(t)\mathbf{s}^H(t)\,(\mathbf{a}^m(\theta))^H - \sigma^{-2}\mathbf{a}^m(\theta)\mathbf{s}(t)\mathrm{z}_m^*(t) \\
& + \hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\,(\hat{\mathbf{a}}^m(t-1))^H - \hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\,(\mathbf{a}^m(\theta))^H \\
& - \mathbf{a}^m(\theta)\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\,(\hat{\mathbf{a}}^m(t-1))^H + \mathbf{a}^m(\theta)\left(\sigma^{-2}\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right)(\mathbf{a}^m(\theta))^H\Big].
\end{aligned}$$

If we complete the square of $\mathbf{a}^m(\theta)$ in the previous expression we will obtain

$$\begin{aligned}
-L(\theta) = {} & \sum_{m=1}^{M} \Big[(\mathbf{a}^m(\theta) - \hat{\mathbf{a}}^m(t))\left(\sigma^{-2}\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right) \\
& \cdot(\mathbf{a}^m(\theta) - \hat{\mathbf{a}}^m(t))^H + \sigma^{-2}|\mathrm{z}_m(t)|^2 + \hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\,(\hat{\mathbf{a}}^m(t-1))^H \\
& - \left(\hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1) + \sigma^{-2}\mathrm{z}_m(t)\mathbf{s}^H(t)\right)\left(\sigma^{-2}\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right)^{-1} \\
& \cdot\left(\hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1) + \sigma^{-2}\mathrm{z}_m(t)\mathbf{s}^H(t)\right)^H\Big], \\
= {} & \sum_{m=1}^{M} (\mathbf{a}^m(\theta) - \hat{\mathbf{a}}^m(t))\left(\sigma^{-2}\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right)(\mathbf{a}^m(\theta) - \hat{\mathbf{a}}^m(t))^H.
\end{aligned} \tag{21}$$

where

$$\begin{aligned}
\hat{\mathbf{a}}^m(t) \stackrel{\text{def}}{=} {} & \left(\hat{\mathbf{a}}^m(t-1)\hat{\mathbf{\Lambda}}_m^{-1}(t-1) + \sigma^{-2}\mathrm{z}_m(t)\mathbf{s}^H(t)\right)\left(\sigma^{-2}\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right)^{-1}, \\
& m = 1,2,\ldots,M.
\end{aligned} \tag{22}$$

Thus the ML estimates $\hat{\mathbf{a}}^m(\theta)$ of $\mathbf{a}^m(\theta)$, $m = 1,2,\ldots,M$ are

$$\hat{\mathbf{a}}^m(\theta) = \hat{\mathbf{a}}^m(t),\ \forall m. \tag{23}$$

The error covariance matrices for these estimates are given by

$$\mathrm{var}\,\{\hat{\mathbf{a}}^m(t)\} = \sigma^2\left[\mathbf{s}(t)\mathbf{s}^H(t) + \sigma^2\hat{\mathbf{\Lambda}}_m^{-1}(t-1)\right]^{-1},\ m = 1,2,\ldots,M, \tag{24}$$

Notice that the change in the error covariance, which is represented by the addition of the outer product $\mathbf{s}(t)\mathbf{s}^H(t)$ inside the matrix inverse, is independent of m. Therefore, if the rows $\hat{\mathbf{a}}^m(t-1), m = 1, 2, \ldots, M$ of $\hat{\mathbf{A}}(t-1)$ were initialized with the same error covariance, say $\sigma^2\hat{\mathbf{\Lambda}}(0)$, then the error covariances for these rows will always be the same. We will assume that this is the case and $\hat{\mathbf{\Lambda}}_m(t-1) = \sigma^2\hat{\mathbf{\Lambda}}(t-1)$, $\forall m$. Hence, the equations which are given in (24) can be rewritten as

$$\text{var}\,\{\hat{\mathbf{a}}^m(t)\} = \sigma^2\left[\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}^{-1}(t-1)\right]^{-1}, \ \forall m, \tag{25}$$

and if we define

$$\hat{\mathbf{\Lambda}}(t) \stackrel{\text{def}}{=} \left[\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}^{-1}(t-1)\right]^{-1}, \tag{26}$$

then we can rewrite equation (25) as follows

$$\text{var}\,\{\hat{\mathbf{a}}^m(t)\} = \sigma^2\hat{\mathbf{\Lambda}}(t), \ \forall m. \tag{27}$$

Moreover, by referring to (22) and using the aforementioned comment of equal covariances of the estimates, we can see that the entire steering matrix can be updated using the following recursive formula

$$\hat{\mathbf{A}}(t) = \left[\hat{\mathbf{A}}(t-1)\hat{\mathbf{\Lambda}}^{-1}(t-1) + \mathbf{z}(t)\mathbf{s}^H(t)\right]\left[\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}^{-1}(t-1)\right]^{-1}. \tag{28}$$

This equation is equivalent to the following expression

$$\hat{\mathbf{A}}(t) = \hat{\mathbf{A}}(t-1) + \left[\mathbf{z}(t) - \hat{\mathbf{A}}(t-1)\mathbf{s}(t)\right]\mathbf{g}(t), \tag{29}$$

where

$$\mathbf{g}(t) \stackrel{\text{def}}{=} \mathbf{s}^H(t)\left[\mathbf{s}(t)\mathbf{s}^H(t) + \hat{\mathbf{\Lambda}}^{-1}(t-1)\right]^{-1}. \tag{30}$$

To obtain a more convenient equation for computing the vector $\mathbf{g}(t)$, we may employ the matrix inversion lemma together with (26) and write

$$\begin{aligned}\mathbf{g}(t) &= \mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t),\\ &= \mathbf{s}^H(t)\left[\hat{\mathbf{\Lambda}}(t-1) - \frac{\hat{\mathbf{\Lambda}}(t-1)\mathbf{s}(t)\mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t-1)}{1+\mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t-1)\mathbf{s}(t)}\right],\\ &= \frac{\mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t-1)}{1+\mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t-1)\mathbf{s}(t)}.\end{aligned} \tag{31}$$

Using the matrix inversion lemma again and (31), we can obtain the following recursive formula for updating the matrix $\hat{\mathbf{\Lambda}}(t)$

$$\hat{\mathbf{\Lambda}}(t) = \hat{\mathbf{\Lambda}}(t-1) - \hat{\mathbf{\Lambda}}(t-1)\mathbf{s}(t)\mathbf{g}(t). \tag{32}$$

By making the following definitions as in [3]

$$\begin{aligned}\mathbf{y}(t) &= \mathbf{s}^H(t)\hat{\mathbf{\Lambda}}(t-1), &(33)\\ \mathbf{u}(t) &= \mathbf{z}(t) - \hat{\mathbf{A}}(t-1)\mathbf{s}(t), &(34)\end{aligned}$$

equations (29), (31) and (32) can then be rewritten in the following simpler form

$$\begin{aligned}\mathbf{g}(t) &= [1+\mathbf{y}(t)\mathbf{s}(t)]^{-1}\mathbf{y}(t), &(35)\\ \hat{\mathbf{A}}(t) &= \hat{\mathbf{A}}(t-1) + \mathbf{u}(t)\mathbf{g}(t), &(36)\\ \hat{\mathbf{\Lambda}}(t) &= \hat{\mathbf{\Lambda}}(t-1) - \mathbf{y}^H(t)\mathbf{g}(t). &(37)\end{aligned}$$

Iterating equations (33), (34), (35), (36) and (37) after each snapshot becomes available from the sensor array is the matrix version of the RLS technique. The vector $\mathbf{g}(t)$ is referred to as the "gain" vector and $\mathbf{u}(t)$ is called the "innovation" vector.

6. Estimating the Signal DOAs

In this section we will derive the formula for obtaining the ML estimate $\hat{\theta}(t)$ of the DOAs θ using $\mathbf{s}(t)$, $\hat{\mathbf{A}}(t)$, and $\hat{\mathbf{\Lambda}}(t)$. To accomplish this, we will assume that an a priori estimate $\hat{\theta}(t-1)$ of θ is available from the previous iteration. However, no assumptions about the statistics of this estimate are presumed.

In order to obtain an estimate of the k^{th} arrival direction $\hat{\theta}_k(t)$, we will examine the M components of the k^{th} steering vector estimate $\hat{\mathbf{a}}_k(t-1)$ of $\hat{\mathbf{A}}(t)$. We will denote the components of this vector by $\hat{\mathrm{a}}_{mk}(t)$, $m = 1, 2, \ldots, M$. Since the RLS algorithm generates approximately unbiased estimates of the regression parameters, we can construct the following nonlinear model for estimating $\hat{\theta}_k(t)$

$$\hat{\mathrm{a}}_{mk}(t) = \mathrm{a}_m(\theta_k) + \epsilon_{mk}(t), \quad m = 1, 2, \ldots, M, \tag{38}$$

where

$$\begin{aligned} E\left[\epsilon_{mk}(t)\right] &= 0, \quad m = 1, 2, \ldots, M, \\ E\left[\epsilon_{mk}(t)\epsilon^*_{nk}(t)\right] &= \delta_{mn}\sigma^2\hat{\mathbf{\Lambda}}_{kk}(t). \end{aligned}$$

The symbol $\hat{\mathbf{\Lambda}}_{kk}(t)$ is used to denote the k^{th} diagonal element of $\hat{\mathbf{\Lambda}}(t)$.

By taking the Taylor's series expansion of $\mathrm{a}_m(\theta_k)$ around the a priori estimate $\hat{\theta}_k(t-1)$ and then truncating the expansion after the linear terms, we will obtain the following approximation of the m^{th} equation which is given in (38)

$$\hat{\mathrm{a}}_{mk}(t) \approx \mathrm{a}_m(\hat{\theta}_k(t-1) + \mathrm{a}_m(\hat{\theta}_k(t-1))\left(-j\psi'_m(\hat{\theta}_k(t-1))\right)\left(\theta_k - \hat{\theta}_k(t-1)\right) + \epsilon_{mk}(t),$$

where $\psi'_m(\hat{\theta}_k(t-1))$ denotes the derivative of the phase delay function taken with respect to θ_k and evaluated at $\hat{\theta}_k(t-1)$. Therefore, the nonlinear equations which are given in (38) can be approximated by the following linear equations

$$\mathrm{r}_{mk}(t) = \mathrm{p}_{mk}(t)\Delta\hat{\theta}_k(t-1) + \epsilon_{mk}(t), \quad m = 1, 2, \ldots, M, \tag{39}$$

where the following definitions have been used

$$\mathrm{r}_{mk}(t) \overset{\text{def}}{=} \hat{\mathrm{a}}_{mk}(t) - \mathrm{a}_m(\hat{\theta}_k(t-1)), \tag{40}$$

$$\mathrm{p}_{mk}(t) \overset{\text{def}}{=} \mathrm{a}_m(\hat{\theta}_k(t-1))\left(-j\psi'_m(\hat{\theta}_k(t-1))\right), \tag{41}$$

$$\Delta\hat{\theta}_k(t-1) \overset{\text{def}}{=} \theta_k - \hat{\theta}_k(t-1). \tag{42}$$

It is easily verified that the linear model which is described by the M equations in (39) have the following least squares solution

$$\hat{\theta}_k(t) = \hat{\theta}_k(t-1) + \left(\mathbf{p}_k^H(t)\mathbf{p}_k(t)\right)^{-1}\mathbf{p}_k^H(t)\mathbf{r}_k(t), \tag{43}$$

where

$$\begin{aligned} \mathbf{p}_k(t) &\overset{\text{def}}{=} \left[\mathrm{p}_{1k}(t), \mathrm{p}_{2k}(t), \ldots, \mathrm{p}_{Mk}(t)\right]^T, \\ \mathbf{r}_k(t) &\overset{\text{def}}{=} \left[\mathrm{r}_{1k}(t), \mathrm{r}_{2k}(t), \ldots, \mathrm{r}_{Mk}(t)\right]^T. \end{aligned}$$

Here $\mathbf{p}_k(t)$ is a M–dimensional column vector and $\mathbf{r}_k(t)$ is a K–dimensional column vector. By referring to the definition that was made in (40), we can see that the vector $\mathbf{r}_k(t)$ can be obtained using the following equation

$$\mathbf{r}_k(t) = \hat{\mathbf{a}}_k(t) - \mathbf{a}(\hat{\theta}_k(t-1)). \tag{44}$$

7. The Recursive Direction Finding Algorithm

In the preceding sections we have derived the equations which are necessary for recursively updating the ML estimates of the signal DOAs. In this section, we will present the complete recursive direction finding algorithm. We will assume that the number of sources K is known and that an a priori estimate $\hat{\theta}(0)$ of θ is available. After first initializing the algorithm using

$$\begin{aligned} \hat{\mathbf{A}}(0) &= \mathbf{A}(\hat{\theta}(0)) \\ \hat{\mathbf{\Lambda}}(0) &= \lambda \mathbf{I} \end{aligned}$$

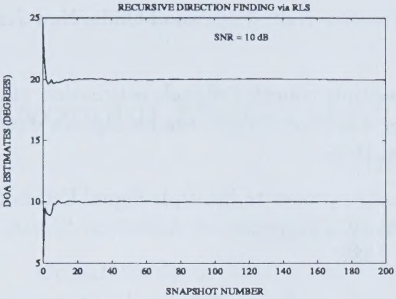
where λ is a positive constant and $\mathbf{I}$ denotes the identity matrix, recursive direction finding can be achieved by iterating the following sequence of equations after each snapshot $\mathbf{z}(t)$ is taken by the sensor array

$$\begin{aligned} \hat{\mathbf{s}}(t) &= \hat{\mathbf{A}}^{+}(t-1)\mathbf{z}(t) \\ \mathbf{y}(t) &= \hat{\mathbf{s}}^{H}(t)\hat{\mathbf{\Lambda}}(t-1) \\ \mathbf{u}(t) &= \mathbf{z}(t) - \hat{\mathbf{A}}(t-1)\hat{\mathbf{s}}(t) \\ \mathbf{g}(t) &= [1 + \mathbf{y}(t)\hat{\mathbf{s}}(t)]^{-1}\mathbf{y}(t) \\ \hat{\mathbf{A}}(t) &= \hat{\mathbf{A}}(t-1) + \mathbf{u}(t)\mathbf{g}(t) \\ \hat{\mathbf{\Lambda}}(t) &= \hat{\mathbf{\Lambda}}(t-1) - \mathbf{y}^{H}(t)\mathbf{g}(t) \\ \mathrm{p}_{mk}(t) &= \mathrm{a}_{m}(\hat{\theta}_{k}(t))\left(-j\psi'_{m}(\hat{\theta}_{k}(t-1))\right),\ m = 1,2,\ldots,M,\ k = 1,2,\ldots,K \\ \mathbf{r}_{k}(t) &= \hat{\mathbf{a}}_{k}(t) - \mathbf{a}(\hat{\theta}_{k}(t-1)),\ k = 1,2,\ldots,K \\ \hat{\theta}_{k}(t) &= \hat{\theta}_{k}(t-1) + \left(\mathbf{p}_{k}^{H}(t)\mathbf{p}_{k}(t)\right)^{-1}\mathbf{p}_{k}^{H}(t)\mathbf{r}_{k}(t),\ k = 1,2,\ldots,K \end{aligned}$$

8. Numerical Results

We conducted three numerical experiments which show that the estimates obtained using this recursive procedure converge to the ones obtained using the less efficient batch processing algorithms. For each experiment, we generated two equally powered uncorrelated narrow-band signals with true arrival directions of 10° for the first signal and 20° for the second signal. The sensor array was linear and consisted of 8 sensors placed a half a wavelength apart. The results of 50 Monte-Carlo runs for each experiment are reported.

In the first experiment, the signal-to noise ratio (SNR) was 10 dB for both signals and the initial guess estimates were 5° and 25° for the first and second signal, respectively. We processed 200 snapshots and the results that we obtained are given in Fig. 1a and Fig. 1b. In Fig. 1a, the DOA estimates after each snapshot is taken by the sensor array are depicted. The corresponding root-mean-square error (RMSE) is given in Fig. 1b and it is compared to the results obtained using the batch processing technique described in [5]. The second experiment was the same as the first, except that the SNR for both signals was lowered to 5 dB. The results that we obtained are given in Fig. 2a and Fig. 2b. In the third experiment, we lowered the SNR to 0 dB. Fig. 3a and Fig. 3b show the results from this experiment. Since the convergence of the recursive estimates to those obtained using batch processing was slower, we processed 300 snapshots in this experiment.



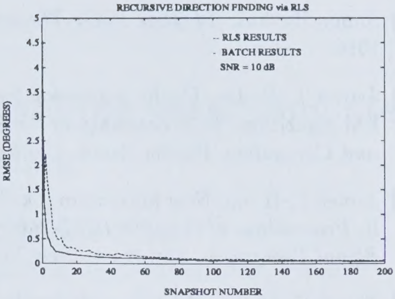
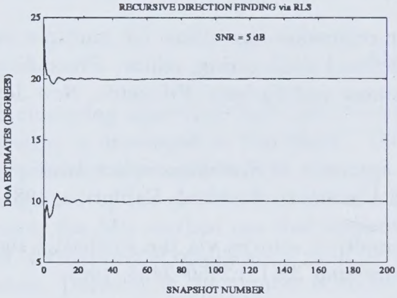


Fig. 1a DOA estimates.

Fig. 1b RMSE results.



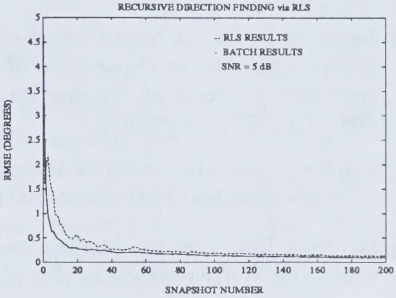
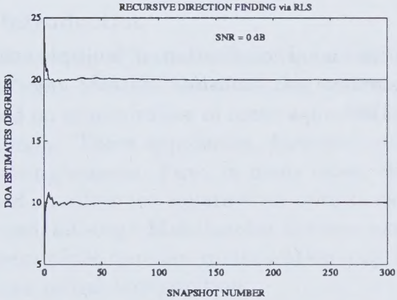


Fig. 2a DOA estimates.

Fig. 2b RMSE results.



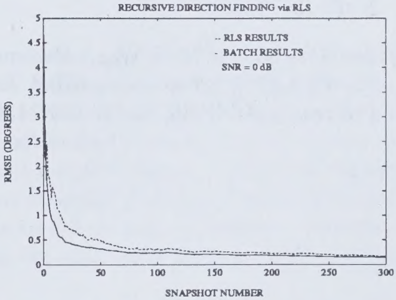


Fig. 3a DOA estimates.

Fig. 3b RMSE results.

9. References

[1] Y. Bresler and A. Macovski. Exact maximum likelihood parameter estimation of superimposed exponential signals in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34, No. 5:1081–1089, 1986.

[2] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, Maryland, 1989.

[3] Simon Haykin. *Adaptive Filter Theory.* Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[4] James T.-H. Lo. Cyclic regression for multiple coherent signals estimation via the EM algorithm. In *Proceedings of the 22nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, 1988.

[5] James T.-H. Lo. New Maximum Likelihood Approach to Multiple Signal Estimation. In *Proceedings of the 1988 IEEE International Conference on Acoustics, Speech, and Signal Processing*, New York, New York, 1988.

[6] James T.-H. Lo. Array sensor localization and calibration by cyclic regression. In *Proceedings of the 1990 International Conference on Acoustics, Speech and Signal Processing*, Albuquerque, New Mexico, 1990. IEEE Signal Processing Society.

[7] James T.-H. Lo. A large class of cyclic regression algorithms for multiple signal direction finding. In Department of Electrical Engineering, editor, *Proceedings of the 1988 Conference on Information Sciences and Systems*, Princeton, New Jersey, 1990. Princeton University.

[8] Ralph O. Schmidt. *A signal subepace approach to multiple emitter location and spectral estimation.* PhD thesis, Stanford University, Stanford, California, 1981.

[9] M. Wax. Detection and localization of multiple sources via the stochastic signals model. *IEEE Transactions on Signal Processing*, 39(11):2450–2456, 1991.

[10] Anthony J. Weiss, Alan S. Willsky, and Bernard. Maximum likelihood array processing for the estimation of superimposed signals. *Proceedings of the IEEE*, 76, No. 2:203–205, 1988.

[11] Ilan Ziskind and Mati Wax. Maximum likelihood localization of multiple sources by alternating projection. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-36, No. 10:1553–1560, 1988.

Maximum Likelihood Clustering with Simulated Annealing*

Kie-Bum Eom and Juha Park
Department of Electrical Engineering and Computer Science
The George Washington University
Washington, D.C. 20052

Abstract

A clustering algorithm based on maximum likelihood (ML) methods and simulated annealing is developed in this paper. The ML clustering method has two advantages over previous approaches. First, the new algorithm searches for the optimal configuration defined by the global maximum of a likelihood function using simulated annealing. Second, the ML method can find nonisotropic clusters as well as clusters of different sizes. Experimental results show good performance of the ML clustering on nonisotropic clusters. Decision on the number of clusters when it is unknown is also discussed in the paper. A stopping rule based on a likelihood ratio test correctly find the true number of clusters in simulated data as well as the Iris data.

1. Introduction

Clustering of multidimensional data is an important subject in many areas of scientific research. Most earlier clustering approaches, noticeably the K-means algorithm, are based on minimization of mean square of inner cluster distance under the circular cluster criterion. These approaches, however, often fail to find correct clusters because of the following reasons. First, in many cases, clusters are nonisotropic. Therefore, algorithms based on minimum square error criteria cannot correctly detect hyperellipsoidal clusters. Second, although Mahalanobis distance can be used to classify nonisotropic clusters [1, 5], deterministic iterative minimization may trap the algorithms in local minima without a proper initial configuration.

Recently, there have been approaches to clustering by simulated annealing [9, 13], which tends to converge to the global minimum. Although algorithms based on simulated annealing require a tremendous amount of computational complexity compared with deterministic minimizing algorithms, recent development in computing power makes these approaches more feasible, especially for solving combinatorial optimization problems [10].

The clustering method developed in this paper differs from previous approaches based on simulated annealing in that it uses simulated annealing to maximize a likelihood function under the Gaussian assumption. Although Bezdeck and Dunn [1] and Gath and Geva [5] use ML methods in their fuzzy clustering algorithms to cluster nonisotropic clusters, deterministic iteration may cause the algorithms to get trapped in local minima.

---

*This work was partially supported by the Office of Naval Research under Grant N00014-91-J-1539.

In most cases, the number of clusters is not known and has to be determined. Thirty so-called "stopping rules" have been evaluated by Milligan and Cooper [12]. Decision by a stopping rule is computationally more attractive than those by performance indices. However, computational attractiveness of the stopping rule is often offset by its inferior performance on finding the true number of clusters.

We develop a stopping rule for selecting the number of clusters using a likelihood ratio test. The likelihood ratio between the number of clusters K and $K+1$ is computed from $K = 1$ until the ratio exceeds a predetermined threshold. The threshold is determined experimentally.

2. Maximum Likelihood Clustering

Suppose that observed samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are independent in the m-dimensional space. Clustering N samples into K clusters is achieved by assigning each sample to one of K classes defined by a cluster distribution. Let $\mathbf{P}$ be a partition of the set of N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ into K subsets $\mathbf{P}_1, \ldots, \mathbf{P}_K$. The number of samples in $\mathbf{P}_k$ is denoted by N_k. Then, the clustering problem becomes that of finding a desirable partition $\mathbf{P}$ that assigns each of N samples into one of K partitions.

If the number of clusters K is known, ML clustering is to find a partition $\mathbf{P}$ that maximizes the likelihood function $p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \mathbf{P}, K)$. Since the samples are assumed to be independent, the likelihood function can be written by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \mathbf{P}, K) = \prod_{k=1}^{K} \left[\prod_{\mathbf{x}_i \in \mathbf{P}_k} p(\mathbf{x}_i | \theta_k) \right] \tag{1}$$

where θ_k is the parameter vector for the probability distribution of the samples in $\mathbf{P}_k$.

Assume that the observed samples are independent Gaussian. Then, mean vector η_k and correlation matrix Σ_k completely define the density function.

$$p(\mathbf{x} | \eta_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{m}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp \left[-\frac{1}{2} (\mathbf{x} - \eta_k)^T \Sigma_k^{-1} (\mathbf{x} - \eta_k) \right] \tag{2}$$

where m is the dimension of the feature space, η_k and Σ_k are the mean vector and the covariance matrix of $\mathbf{P}_k$, respectively.

The likelihood function is obtained by substituting (2) into (1).

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \mathbf{P}, K) = \prod_{k=1}^{K} \frac{1}{[(2\pi)^m |\Sigma_k|]^{\frac{N_k}{2}}} \exp \left[\sum_{\mathbf{x}_i \in \mathbf{P}_k} -\frac{1}{2} (\mathbf{x}_i - \eta_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \eta_k) \right] \tag{3}$$

Then, the log-likelihood function is

$$J(\mathbf{P}, K) = -\sum_{k=1}^{K} \left[\frac{mN_k}{2} \log(2\pi) + \frac{N_k}{2} \log(|\Sigma_k|) + \frac{1}{2} \sum_{\mathbf{x}_i \in \mathbf{P}_k} (\mathbf{x}_i - \eta_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \eta_k) \right] \tag{4}$$

The parameters of the likelihood function in (3) can be estimated by ML methods. Suppose that samples $\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,N_k}$ belong to cluster $\mathbf{P}_k$. Then the log-likelihood function is given by the following equation.

$$\begin{aligned} &\log p(\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,N_k} | \mathbf{P}, K) \\ &\quad = -\left[\tfrac{mN_k}{2} \log(2\pi) + \tfrac{N_k}{2} \log(|\Sigma_k|) + \tfrac{1}{2} \sum_{\mathbf{x}_i \in \mathbf{P}_k} (\mathbf{x}_i - \eta_k)^T \Sigma_k^{-1} (\mathbf{x}_i - \eta_k) \right] \end{aligned} \tag{5}$$

Maximum likelihood estimators, $\hat{\eta}_k$ and $\hat{\Sigma}_k$, are obtained by maximizing the likelihood function in (5) with respect to parameters η_k and Σ_k.

$$\hat{\eta}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \mathbf{x}_{k,i} \tag{6}$$

$$\hat{\Sigma}_k = \frac{1}{N_k - 1}\sum_{i=1}^{N_k} (\mathbf{x}_{k,i} - \hat{\eta}_k)(\mathbf{x}_{k,i} - \hat{\eta}_k)^T \tag{7}$$

To represent the log-likelihood function in terms of the parameter estimates, we first consider a Taylor series expansion of the log-likelihood function given in (5) with respect to parameter estimates. Since the first gradient of the likelihood function in the expansion is zero, the log-likelihood function can be approximated by

$$J(\mathbf{P}, K) \approx -\sum_{k=1}^{K}\left[\frac{mN_k}{2}\log(2\pi) + \frac{N_k}{2}\log(|\hat{\Sigma}_k|) + \frac{1}{2}\sum_{\mathbf{x}_i \in \mathbf{P}_k}(\mathbf{x}_i - \hat{\eta}_k)^T \hat{\Sigma}_k^{-1}(\mathbf{x}_i - \hat{\eta}_k)\right] \tag{8}$$

Maximum likelihood clustering for the given number of clusters K is a partition obtained by maximizing the likelihood function $J(\mathbf{P}, K)$ with respect to all possible partitions of N samples into K partitions.

3. Clustering by Simulated Annealing

For clustering using ML methods, we need to maximize the log-likelihood function $J(\mathbf{P}, K)$ in (8) over all partitions for the given number of partitions K. There are $S(N, K)$ possible ways to partition N samples into K subsets, where $S(N, K)$ is the Stirling number of the second kind [7]. A brute force method requires the evaluation of $J(\mathbf{P}, K)$ in (8) for all the possible partitions, which is computationally expensive even for a small set of data. Of course, numerical approaches to minimization of a cost function, such as the steepest descent method, are available. However, deterministic iterative approaches [7] often do not converge to the global minimum, but give false solutions at local minima.

Although ML clustering approaches in [1, 5] show good performance of nonisotropic clusters, they can suffer from getting trapped in local minima because of deterministic optimization. This suggests that a deterministic optimization algorithm cannot be used to maximize the likelihood function given in (8).

In our work, maximization of the likelihood function in (8) is done by a simulated annealing algorithm, which converges to a global optimum with probability 1 under a proper cooling schedule. Simulated annealing approaches have already been applied to various combinatorial optimization problems [10]. Klein and Dubes [9] apply a simulated annealing technique to clustering and mapping using the sum of square distances criterion. Rose, Gurewitz, and Fox [13] propose a deterministic annealing approach to clustering, where the 'randomness' has been incorporated into the cost function to speed up the cooling process. These two approaches are special cases of the method explained in this paper, since they use the minimum variance criterion, which is a special case of the likelihood function given in (8).

Simulated annealing was proposed by Kirkpatrick *et al.* [8] as a method for minimizing functions having many local minima. The Metropolis algorithm [11] can be used to generate sequences of configurations of a combinatorial optimization problem, such as the clustering problem. In this case, the configurations assume the role of states of a solid while the cost function E and the control parameter T take the roles of energy and temperature, respectively. The simulated annealing algorithm can now be viewed as a sequence of Metropolis algorithms evaluated at decreasing values of the control parameter. Initially, the control parameter T is given a high value, and a sequence of configurations of the combinatorial optimization problem is generated as follows. As in the iterative improvement algorithm a generation mechanism is defined, so that, given a configuration i, another configuration j can be obtained by randomly assigning a single element. The latter corresponds to the small perturbation in the Metropolis algorithm. Let $E_{ij} = E(j) - E(i)$. Then the probability of accepting the configuration j is 1 if E_{ij}, and is $\exp(-E_{ij}/T)$ if $E_{ij} > 0$. Thus, there is a non-zero probability of accepting a new configuration with a higher cost than the current configuration. This process is continued until equilibrium is reached, i.e., until the probability distribution of the configurations approaches the Boltzmann distribution given by the following.

$$\Pr\{\text{configuration} = i\} = \frac{1}{z(T)} \exp\left(-\frac{E(i)}{T}\right) \tag{9}$$

where $z(T)$ is a normalization constant depending on the control parameter T.

Clustering Algorithm by Simulated Annealing:

A simulated annealing algorithm can be used for the maximum likelihood clustering by defining energy function E as the negative function of the likelihood function given in (8).

$$E = \sum_{k=1}^{K} \left[\frac{mN_k}{2} \log(2\pi) + \frac{N_k}{2} \log(|\hat{\Sigma}_k|) + \frac{1}{2} \sum_{x_i \in P_k} (x_i - \hat{\eta}_k)^T \hat{\Sigma}_k^{-1} (x_i - \hat{\eta}_k) \right] \tag{10}$$

where K is the number of clusters, m is the dimension of the feature space, and N_k is the number of elements in P_k. The simulated annealing algorithm for clustering is summarized in the following.

1. Given N data points, initially configure K clusters, $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_K\}$, at random. Compute E for the configuration $\mathbf{P}$. Set temperature T to initial value T_0.
2. At each iteration, decrease T according to a cooling schedule, and
   - (i) Change $\mathbf{P}$ to a new configuration $\mathbf{P}'$ by selecting a sample from one cluster at random and assigning it to a different cluster.
   - (ii) Compute E' for the new configuration $\mathbf{P}'$.
   - (iii) If $E' < E$, accept the change, otherwise accept the change with probability $\exp(-\Delta E/T)$, where $\Delta E = E' - E$.
   - (iv) Repeat (i), (ii), and (iii) until quasi-equilibrium at T is reached.
3. Repeat step 2 until the system is frozen.

The initial temperature T_0 should be chosen sufficiently high such that almost every perturbation is accepted. Otherwise, the algorithm may not reach the global minimum. However, if the initial temperature is chosen too high, unnecessary computations might be performed to reach lower temperature. For different data sets, different values need to be chosen for the initial temperature. To find a proper initial temperature, we first run the algorithm with a randomly selected initial temperature. If the rate of acceptance is less than 99%, the temperature is doubled until the rate is greater than 99%. Similarly, if the acceptance rate is greater than 99%, temperature is lowered by half. When the acceptance rate drops under 99%, the previous value is selected as the initial temperature.

Although the initial temperature is chosen high enough to allow all possible perturbations, a cooling schedule or the rate at which the temperature is lowered must be selected carefully to avoid local minima. Comprehensive work on this subject is presented by van Laarhoven and Aarts [10]. Geman and Geman [6] have shown that, in theory, the global minimum can be reached if the schedule obeys $T \propto 1/\log n$ at nth iteration. However, such a schedule requires an excessive amount of computation time. Since the clustering problem is much simpler that actual annealing process, a faster cooling schedule than $1/\log n$ could be used to achieve optimum. In our experiments, we have found that clustering results are satisfactory with a simple schedule such as $T \propto T_0 a_n$ used in [13], where a is a constant between 0.9 and 1.

In Step 2 of the above algorithm, the number of perturbations, or the length of Markov chain at each temperature, must be large enough to ensure quasi-equilibrium. To avoid excessive computation time, the number of perturbations at each temperature is limited to a certain number. The algorithm stops when the number of consecutive rejections exceeds a given number.

4. Decision on the Number of Clusters

For each K, we can maximize the log-likelihood function $J(\mathbf{P}, K)$ over the partition space $\mathbf{P}$ using the simulated annealing algorithm explained in Section 3. Let $\mathbf{P}^*$ be the partition obtained by maximizing the log-likelihood function $J(\mathbf{P}, K)$. The number of clusters K can be determined by a likelihood-ratio test.

Let H_0 and H_1 be the null and alternative hypotheses, respectively.

H_0: The number of clusters is K.

H_1: The number of clusters is $K+1$.

The likelihood functions for H_0 and H_1 are given by the following equations.

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N | H_0) = p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \mathbf{P}^*, K) \tag{11}$$

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N | H_1) = p(\mathbf{x}_1, \ldots, \mathbf{x}_N | \mathbf{P}^*, K+1) \tag{12}$$

Then, the log-likelihood ratio test for the number of clusters becomes

$$\text{Number of Clusters} \begin{cases} = K, & \text{if } \log \dfrac{p(\mathbf{x}_1, \ldots, \mathbf{x}_\mathbf{N} | \mathbf{P}^*, K)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_\mathbf{N} | \mathbf{P}^*, K+1)} > Q \\ > K, & \text{otherwise} \end{cases} \tag{13}$$

where $Q = \log \frac{\Pr(K+1)}{\Pr(K)}$, and $\Pr(K)$ and $\Pr(K+1)$ are *a priori* probabilities for K and $K+1$ clusters, respectively.

5. Experimental Results

The ML clustering algorithm developed in this paper is implemented and tested with both simulated data and real data. For comparison, the K-means algorithm is also implemented and applied to the same data sets.

Simulated data sets are generated in a similar way used in [2]. However, we modify the algorithm to generate hyperellipsoidal Gaussian clusters. The Iris data set used by Fisher [4] is also used in our experiments. In his work, four measurements were made on each of fifty members of each of three Iris varieties, Iris Setosa, Iris Versicolor, and Iris

Virginica. Iris Setosa is known to be linearly separable from the other two, while there is considerable overlap between Iris Virginica and Iris Versicolor [4].

Experiment with Known Number of Clusters

The clustering algorithm based on simulated annealing explained in Section 3 has been applied to a simulated data set and the Iris data set. In maximization of the likelihood function in (8), the number of clusters is fixed. Good results are obtained by applying a cooling schedule of $T = T_0 c^k$, where k is the number of iterations and c is a constant less than 1 (we used $c = 0.99$). This cooling schedule reduces the temperature much faster than the schedule following $T = c/\log(k+1)$, which guarantees the convergence to the global minimum. In our experiment, however, the simple cooling schedule works very well on test data sets. At each temperature, we set the maximum number of perturbations to five times the sample size to reach a quasi-equilibrium. As the temperature approaches zero, most perturbations are rejected. The simulated annealing procedure stops when the number of consecutive rejections is larger than forty times the number of samples. Even with the simple cooling schedule, the simulated annealing run requires approximately 1,000s on the Silicon Graphics workstation while a single K-means run is completed in less than 1s on the same synthetic data.

During the execution, there are two cases to be considered to ensure that the algorithm will continue to run until the system reaches the global optimum. At each iteration the algorithm estimates the mean and the covariance matrix of each cluster. Since the number of unknown parameters in m-dimensional space is $(m + \frac{m(m+1)}{2})$ for each partition, the number of samples per clusters must be larger than this quantity to reliably estimate the covariance matrix [3]. For this reason, the minimum number of samples in each cluster is set to this value. The other happens when all the samples in a cluster belong to a subspace. Attempted configurations of this type are rejected as well.

Figure 1(a) shows a data set of 80 samples in four clusters, two of which are elongated and close each other. Figure 1(b) shows the best of 400 runs by the K-means algorithm. Since the K-means algorithm uses the minimum variance criterion, it tends to find isotropic clusters. Thus, the K-means algorithm could not correctly find the two elongated clusters as shown in Figure 1(b). In Figure 1(c), all the samples in the test set are classified correctly by the ML method.

The clustering results of the K-means algorithm and the ML method on the Iris data are summarized in Table 1. Both K-means algorithm and ML method by simulated annealing correctly classify the samples in Iris Setosa. However, since there is considerable overlap between Iris Versicolor and Iris Virginica, the K-means algorithm, which uses the minimum variance criterion, does not perform well. Of 150 samples, sixteen samples are misclassified by the K-means algorithm, while only five samples in Iris Versicolor are misclassified by the ML method. Both clustering results obtained here, however, are better than those reported in [9].

Experiment with Unknown Number of Clusters

As we can see in (8), the likelihood function $J(\mathbf{P}^*, K)$ depends on the number of samples and the determinant of the sample covariance matrix of each cluster. $J(\mathbf{P}^*, K)$ generally increases as the number of clusters K increases. Also, the magnitude of log-likelihood ratio $L(K, K+1)$ also depends on the number of samples. Therefore, the threshold Q for the log-likelihood hypothesis test given in (13) should be different for each data set. In our experiments, we found that for a given data set the magnitude of log-likelihood ratio $L(1,2)$ between one cluster and two clusters is the largest of all

$L(K, K+1)$, $K = 1, ..., K_{max}$, unless the data set consists of one cluster. Therefore, we can use a fraction of $L(1,2)$ as the threshold Q, if the magnitude of $L(1,2)$ is larger than that of $L(2,3)$. We used the threshold $Q = L(1,2)/3$, and obtained satisfactory results. If the magnitude of log-likelihood ratio $L(1,2)$ is smaller than that of $L(2,3)$, we decide there is only one cluster in the data set.

Table 2 summarizes the characteristics of test data sets. All five simulated data sets contain clusters in two dimensional space, and the Iris data set contains three clusters in four dimensional space.

The estimation results of the number of clusters are summarized in Table 3. The second row in Table 3 shows the true number of clusters in each data set. Log-likelihood ratios, $L(K, K+1)$'s, are computed and listed in rows 3-8 in the table. As shown in the table, every data set except set 3, which has only one cluster, has larger $|L(1,2)|$ than $|L(2,3)|$. The threshold values are shown in row 9 in Table 3. The estimated cluster numbers are shown in row 10 in the table. In the experiments, the log-likelihood ratio test estimates the number of clusters correctly on all the simulated data and the Iris data. Among other clustering indices for determining the number of clusters, the Davies-Bouldin index and the Calinski-Harabasz index are computed for the Iris data and plotted in Figure 2. A smaller Davies-Bouldin index means better clustering and a larger Calinski-Harabasz index means better clustering. As shown in Figure 2, the Davies-Bouldin index for the Iris data increases monotonically, and the Calinski-Harabasz index decreases monotonically. Although the Calinski-Harabasz index performed best in [12], it failed to detect the true number of clusters in the Iris data.

6. Discussions and Conclusions

In this paper we developed a clustering algorithm based on simulated annealing under Gaussian assumption. Unlike previous simulated annealing approaches, the new approach worked well even on hyperellipsoidal clusters. A stopping rule based on the likelihood ratio test was presented for determining the number of clusters if it unknown.

The ML approach was tested with various simulated data and real Iris data. The new simulated annealing approach performs much better than K-means algorithms on both simulated data and Iris data. The stopping rule seemed to find correct number of clusters while other indices [12] fail.

References

[1] J. C. Bezdek and J. C. Dunn, "Optimal Fuzzy Partitions: A Heuristic for Estimating the Parameters in a Mixture of Normal Distributions," *IEEE Trans. on Computers*, Vol. C-24, pp. 835-838, 1975.

[2] R. C. Dubes, "How Many Clusters are Best? - An Experiment," *Pattern Recognition*, Vol. 20, No. 6, pp. 654-663, 1987.

[3] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, New York, Wiley, 1973.

[4] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Ann. Eugenics 3*, pp. 179-188, 1936.

[5] I. Gath and A.B. Geva, "Unsupervised Optimal Fuzzy Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-11, No. 7, pp. 773-781, 1989.

[6] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Image," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721-741, 1984.

[7] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, N.J., 1988.

[8] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, "Optimization by simulated annealing," *Science*, Vol. 220, pp. 671-680, 1983.

[9] R. Klein and R. Dubes, "Experiments in Projection and Clustering by Simulated Annealing," *Pattern Recognition*, vol. 22, no. 2, pp. 213-220, 1989.

[10] P.J.M van Laarhoven and E.H.L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel, 1987.

[11] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, "Equations of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, Vol. 21, pp. 1087-1092, 1953.

[12] G.W. Milligan and M.C. Cooper, "An Examination of Procedures for Determining the Number of Clusters in Data Set," *Psychometrika*, Vol. 50, No. 2, pp. 159-179, 1985.

[13] K. Rose, E. Gurewitz, G. Fox, "A Deterministic Annealing Approach to Clustering," *Pattern Recognition Letters 11*, pp. 589-594, 1990.

Table 1: Clustering results on Iris data

Original	Classified					
	K-means			Maximum Likelihood		
	Setosa	Versicolor	Virginica	Setosa	Versicolor	Virginica
Setosa	50	0	0	50	0	0
Versicolor	0	48	2	0	45	5
Virginica	0	14	36	0	0	50
Accuracy	89.3 %			96.7 %		

Table 2: Test data sets used in the experiments

Characteristics	Set 1	Set 2	Set 3	Set 4	Set 5	Iris Data
Dimension	2	2	2	2	2	4
No. samples per cluster	20	20	20	100	50	50
No. clusters	4	5	1	2	5	3
Total no. samples	80	100	20	200	250	150

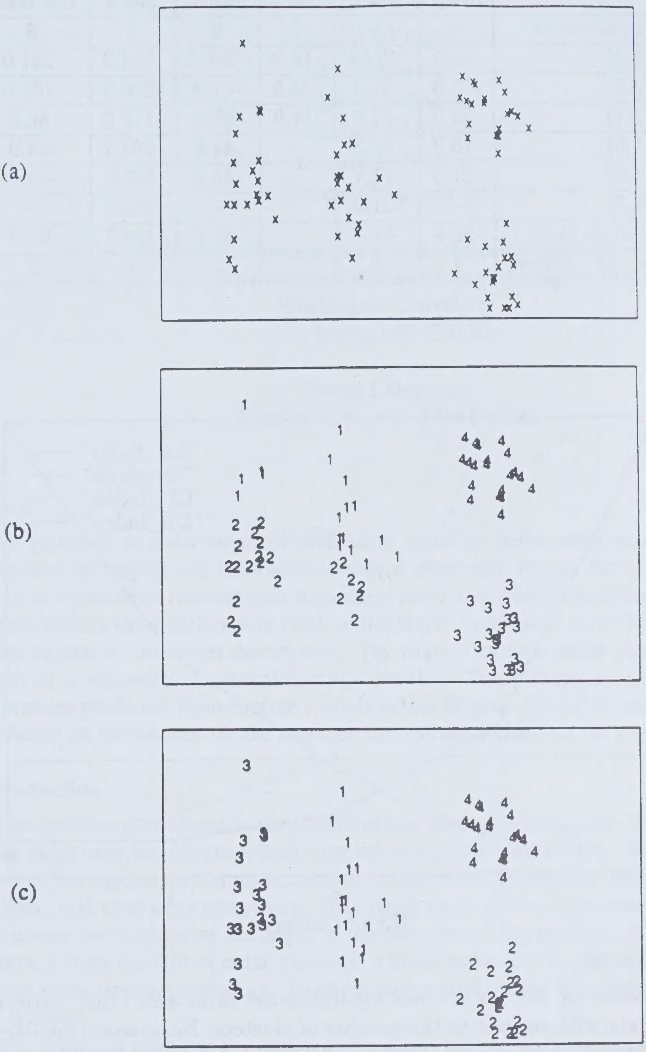


Figure 1: Clustering results on synthetic data set 1. (a) Synthesized data set containing four clusters; (b) Four clusters classified by the K-means algorithm; Many samples are misclassified since the K-means algorithm does not find elongated clusters well; (c) Four clusters classified by the ML method; All samples are correctly clustered.

Table 3: Results on test sets in Table 2

	Set 1	Set 2	Set 3	Set 4	Set 5	Iris Data
True no. of Clusters	4	5	1	2	5	3
L(1,2)	-86.5	-102.4	-18.0	-241.0	-344.2	-261.0
L(2,3)	-70.0	-97.3	-24.5	-61.8	-236.3	-101.6
L(3,4)	-53.9	-52.4	-14.0	-57.0	-172.0	-56.8
L(4,5)	-18.2	-47.3	—	-44.6	-168.7	-44.9
L(5,6)	-21.5	-17.4	—	-41.2	-35.0	-45.4
L(6,7)	—	-21.0	—	—	-27.9	—
Threshold=L(1,2)/3	-28.8	-34.1	—	-80.3	-114.7	-87.0
Estimated no. of Clusters	4	5	1	2	5	3

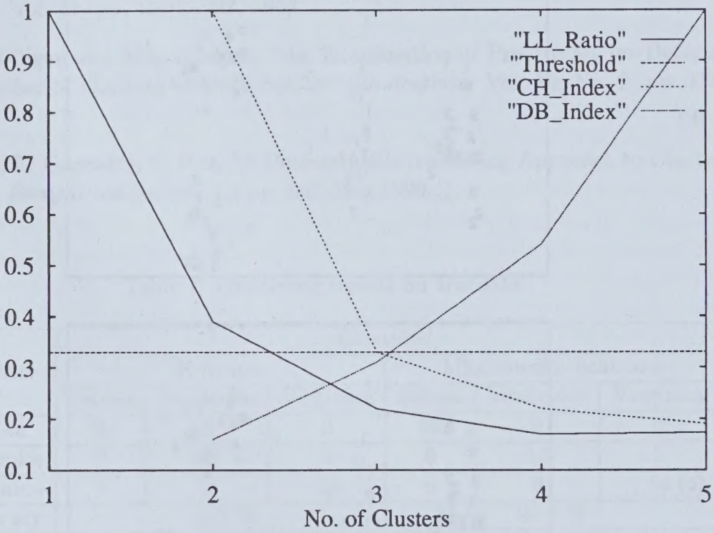


Figure 2: Comparisons of the normalized log-likelihood ratio and other performance indices of the Iris data with respect to the number of clusters. Normalized log-likelihood ratio $L(K, K+1)/L(1,2)$ becomes smaller than normalized threshold (0.33) at $K = 3$, which is the correct number of clusters. Normalized Davies-Bouldin index increases monotonically, while normalized Calinski-Harabasz index decreases monotonically.

Radar Target Recognition Using Shape Models*

Joseph A. O'Sullivan[1]
K. Cecilia Du[1]
Robert S. Teichman[1]
Michael I. Miller[1]
Donald L. Snyder[1]
Vincent C. Vannicola[2]

[1]Electronic Systems and Signals Research Laboratory
Department of Electrical Engineering
Washington University
St. Louis, MO 63130

[2]Rome Laboratory
Griffiss AFB, NY 13441-5700

Abstract

An approach to radar target identification based on deformable templates is presented. Descriptions of targets are incorporated into a stochastic model for radar data. Data are assumed to come from two sensors: a tracking radar and a high resolution radar. The tracking radar collects information over time on the target dynamics. A model of the dynamics is included as part of the target description. The high resolution radar collects information in the form of a sequence of estimated range profiles. These range profiles are compared to range profiles predicted from surface models of the targets. The discrimination between targets is based on likelihoods for the received data parameterized by the target models.

1. Introduction

The problem considered is the identification of targets using data from two sensors. A tracking radar uses waveforms which have relatively low bandwidths. This radar isolates the location of a target to within an uncertainty determined by random measurements, random reflections, and random propagation. This radar may collect a sequence of such data and may estimate the velocity of the target in the direction of the antenna. Using the position of the target, a high resolution radar transmits a chirp pulse and estimates the range profile of the target using stretch processing. From the range profiles and the tracking data, an identification is made.

The theory of "deformable templates" is being specialized to be applicable to the problem at hand. The templates consist of surface descriptions of the targets. These descriptions are of moderate complexity. That is, they do not account for detailed electromagnetic interactions, but do account for the distributed nature of the targets. These templates are presented below. Also discussed is the manner in which range profiles of targets are computed, taking into account a shading mask.

---

*This work was supported by Rome Laboratory under contract number F30602-92-C-0004.

The algorithm proposed for noncooperative target identification (NCTI) may be categorized as identification-by-simulation. This means that estimates of the time-varying target position and orientation are used to obtain a real-time simulation of the target return. This simulated return is compared to the actual return to identify targets. The development of software to accomplish this simulation is progressing. It will be implemented on a DEC/MPP computer in our laboratory. Initial recognition performance using a simplified model shows that the algorithm has much promise.

2. Mathematical Overview

The identification algorithms being pursued are based on statistical models for the data. Several sources of data may be available. Models for each sensor are formulated and models for how the data from the sensors are related are available. This notion of the relationship between sensor data is commonly referred to as "sensor fusion." Here, sensor fusion takes the form of a combination of tracking data and high range resolution data. The tracking data provides prior information on the location and orientation of the target. This prior information contributes directly to the identification routine.

The models for the data from sensors take several forms. Typically these models account for known effects on the radar signal. For high level descriptions of the radar range equation, antenna patterns, antenna gains, etc., see any classical or modern radar text such as [1,2,3,4,5]. The processing performed on the data immediately after reception limits the information extractable. For the moment, an equation for the baseband radar signal available just after demodulating the carrier is derived. This derivation is similar to those in [6] and used in [7,8]. Let $s(t)$ be the complex envelope of the transmitted signal [6]. Let $b(t, \tau)$ be the total reflectivity density of the target at two-way delay τ at time t. That is, $b(t, \tau)$ equals the integral of the reflectivity density on the surface of the target of points at two-way delay τ, at time t. The computation of this integral is addressed in a later section. Assume that there is additive electronics noise in the receiver. Then the available data equals

$$r(t) = \int_0^\infty s(t-\tau)b(t-\tau/2, \tau)d\tau + w(t) , \tag{1}$$

where $w(t)$ is the complex electronics noise. This equation does not directly account for the antenna pattern and the radar range equation. In the problems of interest, the targets may move through a large number of range cells. The signal to noise ratio will vary correspondingly. This may be accounted for in (1) by letting the intensity of the noise vary depending on the bulk range to the target or by including a gain factor in the integral. The antenna pattern for the high resolution radar will be ignored since it is assumed that the high resolution antenna is pointing at the target. This is reasonable since a separate tracking antenna is available and may be assumed to be used to point the second antenna.

Equation (1) governs the data from both the tracking antenna and the high resolution radar. The difference is in the choice of transmitted signal and in the processing performed. For the tracking antenna, the complex envelope $s_1(t)$ has a small bandwidth compared to the target extent. The target looks like a point target at delay τ (with total reflectivity b_{tot}). Typically, the received signal is passed through a matched filter producing the data

$$\int s_1^*(t-\tau')r(t)dt = b_{tot}\rho_1(\tau'-\tau) + \int s_1^*(t-\tau')w(t)dt , \tag{2}$$

where the superscript * indicates complex conjugate, and $\rho_1(\tau)$ is the autocorrelation function for $s_1(t)$

$$\rho_1(\tau) = \int s_1^*(t-\tau)s_1(t)dt .$$

Assuming the antenna pattern is peaked, the location of the target in three dimensional space is determined by the two angles defining the orientation of the antenna and the peak of the output of the matched filter which determines distance. A sequence of such locations determines a target track.

For the tracking algorithm used below, this estimated position is assumed to be random with distribution determined by $\rho_1(\tau)$ and by the antenna pattern. The antenna pattern effectively describes a cone of ambiguity while the autocorrelation function $\rho_1(\tau)$ determines the ambiguity in the range direction.

Other information which may be extracted from the tracking antenna include the bulk velocity of the target in the direction of the antenna. This velocity is obtained using a standard Doppler measurement [3,6]. The use of this in our model is described in section 8.

The high resolution radar is assumed to transmit a sequence of chirp pulses, one of which is

$$s_2(t) = \exp[j2\pi(f_0 t + kt^2/2] , \quad 0 \le t \le T . \tag{3}$$

The bandwidth of this chirp is kT hertz. For the present, it is assumed that for the duration of the pulse, the target may be modeled as being stationary. The range profile for a pulse at time T' equals

$$b(t-\tau/2, \tau) \approx b_{T'}(\tau) , \quad \text{for } t \approx T' . \tag{4}$$

Also, the receiver is assumed to perform "stretch" processing in the following manner (this discussion is based on Wehner's book [3, pp. 115-137]). The stretch processor correlates the received signal with the output of a local oscillator,

$$\exp[-j2\pi(f_0 t + kt^2/2)], \tag{5}$$

over a time period much larger than T. The output of this processor has two terms, a signal term and a noise term.

The noise term is the correlation of $w(t)$ with (5). Since the waveform (5) has a relatively large bandwidth, the resulting noise waveform will be approximately white (assuming $w(t)$ is white). Denote the noise term $\tilde{w}(t)$.

The signal term equals the convolution of $b_{T'}(\tau)$ from (4) with the crosscorrelation between (3) and (5). This crosscorrelation equals $h(t)$

$$h(t) = \int_0^T \exp[j2\pi(f_0\tau + k\tau^2/2)]\exp[-j2\pi(f_0(\tau - t) + k(\tau - t)^2/2)]d\tau \tag{6}$$

$$= \exp[j2\pi(f_0 t - kt^2/2)] \int_0^T \exp[j2\pi kt\tau]d\tau = \exp[j2\pi(f_0 t - kt^2/2)] \frac{1}{j2\pi kt}\left[\exp[j2\pi kTt] - 1\right]$$

$$= \exp[j2\pi(f_0 t - k(t^2 - Tt)/2)]\text{sinc}(kTt) ,$$

where $\text{sinc}(x) = \sin(\pi x)/(\pi x)$. Note that the limits on the first integral in (6) come from the assumption that the local function (5) is nonzero much longer than $s_2(t)$. There are two important points to note about the final expression for $h(t)$ in (6). The first is that the magnitude of $h(t)$ is determined by $\text{sinc}(kTt)$; this function has zeros at $t = n/kT$, for $n \ne 0$, n an integer. The second point is that the phase term has frequency $f_0 + kT/2 - kt$; that is, it is a sinusoid times a chirp.

The output of the stretch processor equals

$$d_{T'}(t) = \int h(t-\tau) b_{T'}(\tau) d\tau + \tilde{w}(t) . \tag{7}$$

Typically a sequence of such waveforms is obtained. Suppose the transmitted signal is

$$\sum_m s_2(t - mT') . \tag{8}$$

The output from the m^{th} signal is

$$d_{mT'}(t) = \int h(t-\tau) b_{mT'}(\tau) d\tau + \tilde{w}(t) . \tag{9}$$

The key to using (9) to identify targets is in having a model for $b_{mT'}(\tau)$ in terms of an underlying template. This model is described below.

The following section presents a model for the computation of $b_{mT'}(\tau)$ from a surface description of the target given the orientation and location of the target. The main salient feature of this model is its characterization of the reflectivity of a target in terms of electromagnetically large patches. The particular mapping from these patches to the reflectivity is rather flexible and may be adapted to account for variations due to such things as different surface materials.

Part of the algorithm computes the likelihood of a given set of data (9) for different target types. This likelihood is determined by the prior on the orientations and positions, the tracking data, and the (stochastic) model for the reflectivity density, $b_{mT'}(\tau)$. Given the reflectivity density, the likelihood for the data (9) is determined by the likelihood for a signal plus the noise $\tilde{w}(t)$. Assuming for the moment that $\tilde{w}(t)$ is white, the computation required is the inner product

$$\log L(d|b) = \frac{1}{N_0} \sum_m \int d_{mT'}(t) \Big[\int h(t-\tau) b_{mT'}(\tau) d\tau \Big] dt . \tag{10}$$

The details of the stochastic model for $b_{mT'}(\tau)$ are discussed below. This model generates a likelihood for the reflectivities $b_{mT'}(\tau)$ given the target type (denoted by i), and given the sequence of positions and orientations at times mT'. These likelihoods are denoted

$$L(b|\mathbf{x}_s, i) \tag{11}$$

where $\mathbf{x}(mT')$ is a vector of positions and orientations and $\mathbf{x}_s$ denotes the sequence of these vectors (indexed by m; the subscript s denotes a sampled version of $\mathbf{x}$).

The likelihood on $\mathbf{x}$ is determined by the dynamics of the target. These dynamics depend on target type and are governed by nonlinear differential equations. A state model for the dynamics given the target type may be written in the general form

$$\dot{\mathbf{x}}(t) = f_i(\mathbf{x}, \mathbf{u}) \tag{12a}$$

$$\mathbf{y}(t) = h_i(\mathbf{x}, \mathbf{v}) . \tag{12b}$$

In these equations, $\mathbf{u}(t)$ denotes a vector of inputs and disturbances, all of which are unobservable; $\mathbf{v}(t)$ denotes a vector of disturbances; and $\mathbf{y}(t)$ denotes a vector of potential measurements. The actual measurements are obtained from the tracking antenna at discrete instants of time (not necessarily uniformly distributed in time). Let t_k denote the sample instants. Then the available measurement data is $\{\mathbf{y}(t_k) : k = 1, 2, \cdots\}$. The likelihood on a sequence $\mathbf{y}_s = \{\mathbf{y}(t_k)\}$ is determined by the distributions on $\mathbf{u}$ and $\mathbf{v}$ in the model (12). In fact, (12a) determines a likelihood $L(\mathbf{x}|i)$ and (12b) determines a likelihood $L(\mathbf{y}|\mathbf{x}, i)$.

Assuming the data $\mathbf{y}_s$ and $\mathbf{b}$ are conditionally independent given $\mathbf{x}$, a joint likelihood may be derived. We have

$$L(d, b, \mathbf{y}_s|\mathbf{x}, i) = L(d|b)L(b|\mathbf{x}, i)L(\mathbf{y}_s|\mathbf{x}, i) \; . \tag{13}$$

Finally, incorporating the likelihood on $\mathbf{x}$ and a prior on i,

$$L(d, \mathbf{y}_s, b, \mathbf{x}, i) = L(d|b)L(b|\mathbf{x}, i)L(\mathbf{y}_s|\mathbf{x}, i)L(\mathbf{x}|i)P_i \; . \tag{14}$$

Remember that in (14), $\mathbf{y}_s$ and d constitute the available data and that the ultimate goal is to determine the most likely target. The functions b and $\mathbf{x}$ may be viewed in two ways. In one view, they are nuisance parameters which should be averaged out. This view is consistent with classic maximum a posteriori (MAP) estimation techniques. This may not be a realistic approach, however. Averaging over these functions may not even make sense. In a second view, these functions are parameters that, if known, would improve the performance of the identification algorithm. This view is consistent with much recent work in estimation theory. Essentially the view is that if the likelihoods are fairly peaked, the actual functions are in the neighborhoods of the MAP estimates for them. Additionally, averaging the functions in the neighborhood of the MAP estimate doesn't increase performance dramatically. Thus, the MAP estimates are used for these functions in place of averaging. Note that the MAP estimates for each of these functions depends on the decision of target type. Denote these estimates

$$\hat{\mathbf{x}}_{MAP}(\mathbf{y}_s, d|i) \quad \text{and} \quad \hat{b}_{MAP}(\mathbf{y}_s, d|\hat{\mathbf{x}}_{MAP}, i) \; .$$

This implies that separate MAP estimates under each hypothesis are required. The target identification algorithm takes the form

$$\hat{i}_{MAP} = \underset{i}{\operatorname{argmax}} L(d, \mathbf{y}_s, \hat{b}_{MAP}(\mathbf{y}_s, d|\hat{\mathbf{x}}_{MAP}, i), \hat{\mathbf{x}}_{MAP}(\mathbf{y}_s, d|i), i) \; .$$

This approach is computationally complex. Presently, methods for implementing it are being explored.

3. Template Description and Radar Reflections

The templates used consist of surface descriptions of the objects of interest. In particular, the surface of an object is partitioned into patches. These patches are electromagnetically large, but physically small. Typically they may be on the order of magnitude of a resolution cell in an image. Each patch has associated with it an area, a normal vector, a vector location from a set of body-centered coordinates, and a type description. The set of patches should form a tiling of the surface of the object. The patch type parameterizes the reflection from that patch through various parts of the model.

In order for this model to be used in practice, templates for the targets of interest are required. Presently, it is unknown how much effort would be required to obtain these models.

In order to compute the received signal, a sum of the reflections off of all patches is formed. This is done in two stages. First, a shading mask is computed. Second, the reflectivity of each point is multiplied by the appropriate time delayed transmitted signal. The validity of this approach for approximating radar reflections is unknown. Detailed electromagnetic models whose patches are smaller than wavelengths of transmitted signals seem inappropriate for our purposes due to their complexity and to their inability to model real world variations (for example, variations of distances to wing tips due to flexing). At the other extreme, having patches much larger than resolution cells seems to be too coarse a model for identification purposes.

The likelihood for the reflectivity density function $b_{mT'}(\tau)$ given the target type, position, and orientation is determined by a statistical model for reflections off patches. Assume the target is at position (x, y, z) in coordinates fixed to the high resolution radar. Assume that patches have four corners whose locations are specified in the body-centered coordinate frame. The i^{th} corner of the k^{th} patch has coordinate location

$$\mathbf{x}_{B,k,i} = \begin{bmatrix} x_{B,k,i} \\ y_{B,k,i} \\ z_{B,k,i} \end{bmatrix}, \quad i = 1, 2, 3, 4.$$

Each corner has associated with it a vector normal to the surface,

$$\mathbf{n}_{k,i} = \begin{bmatrix} n_{x,k,i} \\ n_{y,k,i} \\ n_{z,k,i} \end{bmatrix}, \quad i = 1, 2, 3, 4.$$

The body centered coordinate frame is rotated with respect to the radar axes by angles (ϕ, θ, ψ) (roll, pitch, and yaw). The corresponding rotation matrix is

$$\mathbf{R}(\phi, \theta, \psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The position vectors of the corners of the patches are determined by first translating them by (x, y, z), then rotating them with $\mathbf{R}(\phi, \theta, \psi)$

$$\mathbf{x}_{I,k,i} = \mathbf{R}(\phi, \theta, \psi)[\mathbf{x}_{B,k,i} + (x, y, z)^T].$$

The normal vectors for the corners are obtained by rotating them with $\mathbf{R}(\phi, \theta, \psi)$.

The reflectivity of the target is determined by the reflectivity of individual patches. Initially, it is assumed that the patches are electromagnetically noninteracting. Additionally, patches which are not directly illuminated are assumed not to contribute to the reflectivity. This second observation leads to the need to compute the shading mask. In order to clarify the following discussion, let $g_{k,i}$ be the shading mask for the i^{th} corner of the k^{th} patch for the given target orientation (the dependence of $g_{k,i}$ on (ϕ, θ, ψ) is suppressed for notational convenience). Note that $g_{k,i} \in \{0,1\}$, with $g_{k,i} = 0$ corresponding to a shaded corner.

The reflection off an illuminated patch may be computed in several ways. Ideally, an approximation of the integral across the patch of the incident signal times an appropriate reflectivity density function should be computed. The reflectivity density function must include effects due to the properties of the surface material and the angle of the incident signal with respect to the normal vectors. Note that as the angle of the incident wave varies, the distances of individual points on the patch from the transmitter vary. Thus the result of the integral across the patch must vary with angle of incidence.

Our model approximates the integral of the transmitted signal times the reflectivity density across a patch by a sum of four terms due to smaller regions located at the corners of the patch. At each corner, a region whose surface is in the plane perpendicular to the normal vector is formed. This region has area determined by the two vectors from the corner location to the adjacent corners. For example, suppose that for patch k, the four corners are indexed sequentially around the patch. Then two vectors associated with the corner are

$$\mathbf{v}_1 = \mathbf{x}_{l,k,2} - \mathbf{x}_{l,k,1} \quad \text{and} \quad \mathbf{v}_2 = \mathbf{x}_{l,k,2} - \mathbf{x}_{l,k,3} \; .$$

The area associated with this corner equals the area of the parallelogram whose sides are one half of these vectors. Let × denote vector cross-product and $A_{k,2}$ denote the area, then

$$A_{k,2} = \frac{1}{4} \|\mathbf{v}_1 \times \mathbf{v}_2\| \; ,$$

where $\|\cdot\|$ denotes euclidean norm.

The reflectivity is computed by summing the contributions of the corners of patches which are illuminated. Note that the first coordinate of $\mathbf{x}_{l,k,i}$, $x_{l,k,i}$, equals the range from the transmitter to the corner of the patch. This distance contributes a deterministic phase to the reflectivity,

$$\exp[j4\pi f_0 x_{l,k,i}/c] \; ,$$

where f_0 is the carrier frequency and c is the speed of propagation. The bulk distance to the target is usually not know to within a wavelength, so there is a random phase component common to all reflectivity terms. Denote this phase η and assume the probability density function for η is known (usually assumed to be uniform on $[0,2\pi]$). For each corner, there is an additional phase uncertainty term due to the result of integrating the reflectivity across the patch. Denote this phase term $\theta_{k,i}$. The model for $\theta_{k,i}$ is important. At one extreme, it may be modeled as being uniformly distributed, independent from corner to corner. At the other extreme, it may be modeled as being equal to 0. In between, models may incorporate random dependencies between the phases. For now, it is assumed that $\theta_{k,i}$ are independent, identically distributed (i.i.d.) random variables with probability density function $p_\theta(\theta)$, but this can be modified if desired.

The amplitude of the reflectivity is determined by the angle between the incident radar wave and the normal vectors and the area of the regions. Let $a_{k,i}$ be the amplitude, and let $\mathbf{e}_1 = [-1,0,0]^T$ be the direction of propagation, then the incidence angle equals

$$\xi_{k,i} = \cos^{-1}(\mathbf{e}_1 \cdot \mathbf{R}\mathbf{n}_{k,i}) \; ,$$

where $\mathbf{R}$ is the rotation matrix. The amplitude of the reflection equals

$$a_{k,i} = b g_{k,i} f(\xi_{k,i}, A_{k,i}) \; ,$$

where b is a constant and $f(\cdot)$ is an appropriate function. The parameter b determines the average reflectivity. Choice of $f(\cdot)$ models the directional dependence of the reflection. For example,

$$f(\xi_{k,i}, A_{k,i}) = A_{k,i} \cos \xi_{k,i}$$

models a cosine-squared energy dependence on the incidence angle. A second example is

$$f(\xi_{k,i}, A_{k,i}) = A_{k,i} \frac{J_1(2\pi \sin(\xi_{k,i}) r_{k,i}/\lambda)}{2\pi \sin(\xi_{k,i}) r_{k,i}/\lambda} \; ,$$

where $J_1(\cdot)$ is a Bessel function of the first kind, $r_{k,i}$ is found such that $\pi r_{k,i}^2 = A_{k,i}$, and λ is the wavelength of the carrier. This choice corresponds to the antenna pattern of a circular aperture centered at the corner, with area equal to the area of the parallelogram.

Finally, suppose that at time mT' the target is at distance (x, y, z) and oriented at angle (ϕ, θ, ψ). Then the reflectivity density $b_{mT'}(\tau)$ given by the model equals

$$b_{mT'}(\tau) = e^{j\eta} \sum_k \sum_{i=1}^{4} a_{k,i} \exp[j(4\pi f_0 x_{I,k,i} + \theta_{k,i})]\delta(\tau - 2x_{I,k,i}/c) \,,$$

where $\delta(\cdot)$ is a Dirac delta function. The model which led to this equation defines the likelihood function $L(b|\mathbf{x}_s, i)$ described above.

One potential improvement in the model described above would be to assume the parallelogram at each corner is centered not at the corner, but shifted from the corner of the patch so that one corner of the parallelogram is at the corner of the patch. This corresponds to adding to the corner location $\frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$ with $\mathbf{v}_1$ and $\mathbf{v}_2$ defined above.

Other improvements include modeling the time dependent behavior of $\theta_{k,i}$. For example, the probability density function of $\theta_{k,i}$ at time mT' given the value of $\theta_{k,i}$ at time $(m-1)T'$ may be given.

4. Target Identification

Target identification is achieved by maximizing the likelihood of a target given the data. In equation (14), the functions which are not averaged out are estimated while simultaneously estimating target type. For example, in (14), b corresponds to the reflectivity data. Estimates for the functions $b_{mT'}(\tau)$ are generated, and these estimates are plugged into the likelihood function and used to evaluate the likelihood of target i. Thus, the target identification algorithm simulates range profiles. This justifies calling this procedure identification-by-simulation.

Some preliminary work has been done to evaluate part of the target identification strategy. Since the computation of $L(d|b)$ involves the inner products shown in (10), a simulation to show identification using just inner products has been performed. The simulation consists of the identification of the orientation and location of a given airplane from a noisy (additive Gaussian noise) image. This simulation has been implemented on three different massively parallel (SIMD) computers and the results are encouraging. In the future, we will connect the inner product computation done here with the other terms in the likelihood to compute (14) and use that for identification.

5. Conclusions and Future Work

The models for target tracking and dynamics are based on standard dynamic models [9]. The main difficulty is in deriving the likelihoods $L(\mathbf{x}|i)$. In order to make the method more practical, it may be necessary to use simplified models for the dynamics. Discrimination may be achieved, in part, by hypothesis testing on the dynamics. At the very least, this may yield an efficient method for reducing the search space required in the template matching stage.

Despite the preliminary nature, some of the results are very encouraging. The real-time demonstration of least-squares matching of airplane images is one reason for this excitement. The best match is determined with high probability. This demonstration must be combined with the computations of target range profiles in order to get a more realistic scenario. In addition, this demonstration has ignored some sensor characteristics which will be included.

References

1. M. I. Skolnik, *Introduction to Radar Systems, Second Edition,* McGraw-Hill, New York, 1980.

2. F. E. Nathanson, J. P. Reilly, and M. N. Cohen, *Radar Design Principles: Signal Processing and the Environment,Second Edition,* McGraw-Hill, New York, 1991.
3. D. R. Wehner, *High Resolution Radar,* Artech House, Dedham, MA., 1987.
4. D. L. Mensa, *High Resolution Radar Imaging,* Artech House, Norwood, MA, 1981.
5. A. W. Rihaczek, *Principles of High-Resolution Radar,* Peninsula Publishing, Los Altos, CA, 1985.
6. H. L. Van Trees, *Detection, Estimation, and Modulation Theory, Part III,* John Wiley and Sons, New York, 1971.
7. D.L. Snyder, J.A. O'Sullivan, and M.I. Miller, "The Use of Maximum-Likelihood Estimation for Forming Images of Diffuse Radar-Targets from Delay-Doppler Data," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 536-548, 1989.
8. P. Moulin, J. A. O'Sullivan, and D. L. Snyder, "A Method of Sieves for Multiresolution Spectrum Estimation and Radar Imaging," *IEEE Transactions on Information Theory,* vol. IT-38, pp. 801-813, March 1992.
9. B. Friedland, *Control System Design: An Introduction to State-Space Methods,* McGraw-Hill, New York, 1986.

Wald's SPRT Applied to Speaker Verification

Michael A. Lund and C.C. Lee
Northwestern University
Department of Electrical Engineering and Computer Science
McCormick School of Engineering and Applied Science
Evanston, IL, 60208-3118, USA.
708-491-7375

Abstract

The problem of speaker verification is separated into a signal processing front-end which produces a test statistic, and a decision-theoretic verification section. It is shown that for a simple front-end, the test statistic is nearly Gaussian and that verification algorithms based on these Gaussian statistics perform quite well. Based on the Gaussian assumption, we derive the Sequential Probability Ratio Test (SPRT) and the modified SPRT for speaker verification. We demonstrate the need for robust processing techniques in making the true speaker/imposter decision, and show that the modified SPRT promises to be a good initial step in that direction. We demonstrate on a speech database that the SPRT and modified SPRT behave as expected, and show that significant improvement in speaker discrimination is achieved. In particular, the modified SPRT is shown to achieve a 5% error rate in a speech independent mode.

1 Introduction

The problem of speaker verification can be formulated as a binary hypothesis testing problem, in which the input speech is attributed either to the claimed speaker or to the class of all other speakers in the speaker pool. Generally, the input speech is reduced to a set of parameters which is assumed to represent the speaker space, and this set is in turn reduced to a single decision statistic which is used in verifying the identity claim. Most previous work has focused on the front end of the speaker verification system, i.e. on the parameter set used to represent the speaker space, and the distance measure used to generate the test statistic. Little attention has been paid to the final decision making process, however.

Errors are usually classified as false rejections or false acceptances. Because the costs associated with the two types of errors are dependent on the application, most researchers in the field have focused their attention on minimax detection schemes which yield an equal error rate for the two types of errors. The challenge lies in determining (a priori) a decision algorithm which will perform as well as the theoretical (a posteriori) equal error rate. In other words, it is easy to determine the optimal threshold for making an equal error rate decision after performing experiments on the test data, but far more difficult to determine a useful threshold based on the training data alone.

Burton [1] assumed a Gaussian distribution for his test statistic, and determined the equal error rate decision threshold based on the Gaussian parameters derived from the training data. Burton's referee attributed the bias in Burton's results (in favor of a higher number of false rejections) to the failure of the Gaussian assumption; our results indicate, however, that the Gaussian assumption leads to nearly equal error rate performance when testing against the training data, while severe bias is encountered when testing against data from later verification sessions. This bias is removed if the decision threshold is based upon the Gaussian parameters of the test data rather than those of the training data. Thus the problem lies not in the Gaussian assumption, but rather in a lack of robustness with respect to uncertainty in the test data statistics.

Once we accept the Gaussian assumption, it is straightforward to formulate the decision algorithm as a likelihood ratio test, and to derive Wald's sequential probability ratio test (SPRT) for speaker verification. Although sequential testing has been used previously (see for example Naik and Doddington [2]), it has usually been within a heuristic framework. Here we derive the speaker verification SPRT on a detection theoretic basis, and evaluate it's performance on a small speech database.

In the following sections, we describe the speaker verification front-end, the hypothesis test formulation, and the speaker verification SPRT. Then we address the issue of test robustness and discuss our system performance.

2 Speaker Verification Front-End

In order to focus attention on the final decision algorithm, the front-end of the speaker verification system was designed to be as simple as practicable, while still generating a useful test statistic. Thus we chose to implement a text-independent system based on the constrained vocabulary consisting of natural number combinations, such as "Twenty-seven Thirty-five Seventy-nine". DFT-cepstral coefficients were vector quantized using a single VQ codebook for each speaker, and a cepstrally weighted Euclidean distance measure [3] was used to calculate the VQ distortion, which served as our test statistic.

The input speech was recorded over local telephone lines, and was extracted from the noise background using the endpoint detection algorithm of Rabiner and Sambur [4]. The digitized speech samples were high-frequency emphasized as in [5], and 12 DFT-based cepstral coefficients were determined for each 45 ms frame of speech, with 2/3 overlap between frames.

In the training phase, the Linde, Buzo, Gray algorithm [6] was used to design a VQ codebook for the cepstral coefficients of each speaker. The cepstral weighting of the Euclidean distance measure was based on the intra-speaker variance of the cepstral coefficients, averaged over the entire speaker pool.

In the testing phase, the cepstral distance measure was calculated for the input speech using the VQ codebook corresponding to the claimed speaker identity.

In order to overcome problems related to unequal recognition performance of different digits (e.g., "nine" is far better in differentiating between speakers than "six"), the verification utterances were chosen in a random manner.

The training data for each speaker consisted of 24 utterances recorded during a single session, while the test data was generated by placing calls to the enrolled subjects at random times in the time period subsequent to the training session. Some subjects provided data for a period of more than six months, and all subjects participated for at least one month. The database was collected in a realistic speaker verification setting, with a commercially available verification system providing the subjects with immediate feedback as to the success of each verification attempt. The time period spanned by the test database was long enough to produce significant variations in the statistical parameters for all speakers.

3 Hypothesis Test Formulation

Define

$$\begin{array}{lll} H_0: & \text{true speaker} & d_i = \mu_0 + n_{0i}, \quad n_{0i} \sim N(0, \sigma_0^2) \\ H_1: & \text{imposter} & d_i = \mu_1 + n_{1i}, \quad n_{1i} \sim N(0, \sigma_1^2), \end{array} \tag{1}$$

where d_i is the test statistic for the i^{th} utterance, μ_0 is the mean of the test statistic over all the training data for the true speaker, μ_1 the mean of the test statistic over all the training data for the remainder of the speaker pool, and σ_0^2 and σ_1^2 the intra- and inter-speaker variances of the test statistic. $N(0, \sigma^2)$ represents the Gaussian distribution with mean 0 and variance σ^2.

Notice that the true speaker and imposter distributions will be different for each speaker, as there will be differences in how well each speaker's VQ codebook differentiates between speakers.

4 The Gaussian Assumption

Although our effective framelength is 15 ms, the 30 ms overlap between adjacent frames may introduce some correlation between frames. Specifically, for each effective 15 ms frame, we calculate

$$X_k = \sum_{i=1}^{12} \frac{1}{\sigma_i^2}(c_{ik} - \mu_{ij})^2 \tag{2}$$

where c_{ik} is the i^{th} cepstral coefficient for the k^{th} frame of speech, and μ_{ij} is the i^{th} component of the j^{th} codebook vector, where $j = \arg\min_l \sum_{i=1}^{12}(1/\sigma_i^2)(c_{ik} - \mu_{il})^2$.

The overlap of frames will cause X_{k-2},X_{k-1},X_{k+1}, and X_{k+2} to be correlated with X_k, making a central limit theorem (C.L.T.) argument problematic. However, if we partition the sequence $\{X_k\}$ into the three sub-sequences $\{X_{0k}\}$, $\{X_{1k}\}$, and $\{X_{2k}\}$, defined as

$$\{X_{ik}\} = \{X_i, X_{i+3}, X_{i+6}, \ldots\} \tag{3}$$

and further define

$$S_{in} = \sum_{k=1}^{n/3} X_{ik}, i = 0, 1, 2 \tag{4}$$

then the C.L.T. guarantees that each of the S_{in} will converge to a Gaussian distribution as $n \longrightarrow \infty$.

Our test statistic is $d_i(n) = 1/n \sum_{i=0}^{2} S_{in}$. For large n, $d_i(n)$ is the sum of three Gaussian random variables, and thus Gaussian. Since in our case n is equal to the number of 15 ms frames in an utterance (usually at least 100 frames), it is safe to assume that the test statistic is approximately Gaussian.

5 The Speaker-Verification SPRT

Given the hypothesis testing formulation of equation (1), and the Gaussian assumption of the previous section, we can easily derive Wald's SPRT [7] for this problem.

The likelihood ratio for the i^{th} utterance can be expressed as

$$L(d_i) = \frac{\sigma_0}{\sigma_1} \exp\left\{\frac{\sigma_1^2(d_i - \mu_0)^2 - \sigma_0^2(d_i - \mu_1)^2}{2\sigma_0^2\sigma_1^2}\right\}, \tag{5}$$

and the likelihood ratio for a sequence of n i.i.d utterances is

$$L_n(\mathbf{d}) = \prod_{i=1}^{n} L(d_i), \mathbf{d} = (d_1, \ldots, d_n). \tag{6}$$

The i.i.d. assumption is valid because of our random choice of utterances.

The likelihood ratio test for a single utterance is given by

$$L(d_i) \begin{array}{lll} > & \tau & \text{choose} \quad H_1 \\ \leq & \tau & \text{choose} \quad H_0 \end{array} \tag{7}$$

and the SPRT is implemented as

$$L_n(\mathbf{d}) \begin{array}{lll} > & B & \text{choose } H_1 \\ \epsilon & [A, B] & \text{get another utterance} \\ < & A & \text{choose } H_0 \end{array} \tag{8}$$

Using the Wald approximations $B = (1 - \gamma_d)/\alpha_d$ and $A = \gamma_d/(1 - \alpha_d)$, where α_d is the desired false rejection rate and γ_d the desired false acceptance rate, we are guaranteed that the actual performance will be bounded by α_a and γ_a, such that

$$\begin{aligned} \alpha_a &\leq \frac{\alpha_d}{1 - \gamma_d} \\ \gamma_a &\leq \frac{\gamma_d}{1 - \alpha_d}. \end{aligned}$$

This performance guarantee assumes that the Gaussian assumption holds and that the model parameters are known without uncertainty, and thus must be viewed suspiciously in the context of speaker verification.

6 Robustness

It is an unfortunate fact that a speaker's voice characteristics will change over time. Even given a set of parameters which perfectly capture the essential character of the speaker space at any particular instant, these parameters will always be sub-optimal at a later time. Thus, in order for a speaker-verification system to perform well over a prolonged period, there is a need for a mechanism whereby the speaker characterization is allowed to adapt over time, or for a decision algorithm which is robust with respect to some temporal variation of the parameters. This paper will address the second of these approaches.

Our experiments indicate that the training data adequately captured the statistical properties for our imposter test data, but that the training data statistics were not sufficient to characterize the test data statistics for the true speaker case. In particular, the mean distortion for each speaker was invariably greater for the test data, and the variance of the distortion also demonstrated significant variability.

Initially we consider only the single utterance case.

As a first attempt at achieving greater system robustness with respect to changes in the model statistics, a standard epsilon contamination model [7] is applied to the speaker-verification hypothesis test. We assume the imposter model to be known completely, and the true speaker model to be

$$H_0' : d_i \sim (1 - \epsilon)N(\mu_0, \sigma_0^2) + \epsilon M_0, \tag{9}$$

where M_0 is some unspecified contaminating density. Under this assumption, our goal is to determine the minimax decision rule for the hypothesis test H_0' vs. H_1, optimized over the class of all possible contaminating densities M_0. As in Poor [7], the solution to the minimax problem is determined as the likelihood ratio for the worst case contaminating distribution. This yields the previous likelihood ratio test, but modified such that

$$L_r(d_i) \begin{array}{lll} > & \tau & \text{choose} \quad H_1 \\ \leq & \tau & \text{choose} \quad H_0 \end{array} \tag{10}$$

where $L_r(d_i)$ is a limited likelihood ratio function,

$$L_r(d_i) = \begin{array}{ll} \frac{1}{1-\epsilon} L(d_i) & \text{if} \quad L(d_i) < c'' \\ \frac{c''}{1-\epsilon} & \text{if} \quad L(d_i) \geq c'' \end{array} \tag{11}$$

where c'' is chosen to satisfy the constraint

$$(1-\epsilon) P_0\{L(d_i) < c''\} + \frac{1}{c''}(1-\epsilon) P_1\{L(d_i) \geq c''\} = 1 \tag{12}$$

and $L(d_i)$ is defined as before.

Experiments showed this model to be far too conservative, yielding unacceptably high error rates. This model is expected to be more appropriate in a verification system with a more sophisticated front-end, such that ϵ would be much smaller.

In order to determine the nature of the temporal variation of the statistical parameters, a series of experiments was run, in which the VQ codebooks generated using the training data were used in conjunction with the statistics generated from the test data. This experiment was designed in order to test how well the Gaussian assumption performed given appropriate model characteristics. The results of these experiments showed that (1) the Gaussian assumption is still appropriate for use on the test data, and (2) in order to use the Gaussian assumption effectively, an algorithm for adapting the decision threshold must be applied.

Unfortunately, the degree of variation between the training and test statistical parameters was not uniform over the speaker pool. The temporal stability of the parameters from one session to another does not appear to be correlated strongly with verification performance on the training data set. In our small speaker pool, there were several speakers who had above average verification performance on the training data, but whose degree of parameter variation was quite severe. On the other hand, a few of the poor performers had quite stable speech parameters.

More specifically, we found that the mean distortion for the test data was greater than the mean distortion for the training data for each speaker in our database. Since variations in the assumed mean of our Gaussian distribution appear to be far more damaging than small variations in the assumed variance, we are led to the following simplification of the general epsilon-contamination model:

Rather than apply a minimax procedure over the class of all ϵ-contaminated mixtures, we can implement a simpler system, in which we assume that the test data is still adequately described by a simple Gaussian distribution, but with uncertainty in the mean.

The same minimax procedure as above is applied to the case

$$H_0'' : d_i \sim N(\mu_0', \sigma_0^2) \tag{13}$$

where μ_0', $\mu_0 \leq \mu_0' \leq (1+\epsilon)\mu_0$, is an unknown parameter. The worst case distribution in this case is $N((1+\epsilon)\mu_0, \sigma_0^2)$, yielding the original likelihood ratio expression of Equation (5), with μ_0 replaced by $(1+\epsilon)\mu_0$.

We are left with the non-trivial problem of determining the value of ϵ for each speaker. Unfortunately, no single value of ϵ can be found which will perform uniformly well over the entire speaker pool. Since we have a high degree of confidence in our imposter statistics, we will use a Neyman-Pearson based approach to determine ϵ.

In particular, we consider the likelihood-ratio test based on a single utterance, fix the probability of false acceptance based on this test at a suitable level, and determine the combination of μ_0 and τ which satisfies both the constraint that the probability of false acceptance not exceed our chosen level, and the constraint that we achieve an equal error rate. In other words, we fix the minimum level of performance that we will tolerate, and determine the greatest value of ϵ that will allow us to achieve this performance goal. If the actual value of ϵ is less than this value, our performance will exceed our goal. If the actual value of ϵ is larger, then no decision strategy based on a single utterance will be able to meet our performance requirements.

At this point a severe problem with the optimal likelihood ratio must be addressed. In general, σ_0 will be significantly smaller than σ_1. Because of the strength of the tail of the Gaussian distribution, this disparity in the variances will lead to two decision regions for the imposter hypothesis. The first region will correspond to our desired region, while the second region will include data points which fall significantly below our assumed μ_0. Given the nature of our data, it is highly unlikely that such data points would be caused by an imposter.

In order to compensate for this problem, we must modify our likelihood ratio test. For a single utterance, this modification is achieved by manipulating the likelihood ratio test into the following form:

$$(d_i - k)^2 \begin{array}{lll} > & x & \text{choose } H_1 \\ \leq & x & \text{choose } H_0 \end{array} \tag{14}$$

where

$$k = \frac{\mu_0 \sigma_1^2 - \mu_1 \sigma_0^2}{\sigma_1^2 - \sigma_0^2} \tag{15}$$

The positive root of this equation corresponds to the desired decision region, and we are left with the modified likelihood ratio test

$$d_i - k \begin{array}{lll} > & +\sqrt{x} & \text{choose } H_1 \\ \leq & +\sqrt{x} & \text{choose } H_0 \end{array} \tag{16}$$

assuming $\sigma_0 < \sigma_1$.

This test is useless for use in the SPRT, however, because we cannot extend the test of equation 16 to the case of multiple observations. In order to use the sequential test we must make the further simplification of mapping the decision statistic to a discrete random variable. Using the modified likelihood ratio test above to determine μ_0 and x such that our equal error rate is 0.15, we know that

$$\begin{gathered} Prob(d_i < \mu_0 | H_0) = 0.5 \\ Prob(\mu_0 \leq d_i < \sqrt{<x} | H_0) = 0.35 \\ Prob(\sqrt{x} \leq d_i | H_0) = 0.15 \end{gathered} \tag{17}$$

and similarly under H_1. Using these probabilities, we define the discrete decision statistic d' and use the corresponding discrete likelihood ratio. This discretized test will be called the modified SPRT.

7 Results

Experiments were conducted for a speaker pool consisting of six male and six female speakers. Imposter testing was performed using only same-sex speakers, using three VQ codebook sizes: 16, 32, and 64. In each case, epsilon was determined for a desired probability of error of 0.15, and α_d and β_d were set at 0.05. The modified SPRT was compared to the robust likelihood ratio test based on a single utterance, and for the robust likelihood ratio test based on a concatenation of two utterances. On average, the modified SPRT needed 2.8 utterances to reach a decision for an imposter, and needed 2.1 utterances to reach a decision for the true speaker.

Figure 1 shows the mean error rate for false rejections. For each code-book size, the single utterance performance is significantly less than 15%, indicating that a smaller upper bound on acceptable performance could have been chosen. The sequential test outperforms the single utterance test, as expected, and performs at about the same level as the two utterance test. The performance advantage of the modified SPRT becomes clear however, when we consider the results for the probability of false acceptance, as shown in Figure 1. Here the modified SPRT outperforms the two utterance test by nearly 10%. This is to be expected, as the two utterance test was designed based upon an assumed 15% false acceptance rate, while the modified SPRT was designed to achieve a 5% equal error rate.

Figure 3 and Figure 4 show the variance over the speaker pool for the false rejection and false accpetance rate, respectively. In each case, the sequential test is seen to perform more consistently than the single and two utterance tests. It should be noted that the tests based on the training statistics alone, without any form of adaptation, do not yield any acceptable results. In fact, the rise in μ_0 is usually significant enough to cause a probability of false rejection approaching 1.0.

8 Conclusions

In this paper we have used the SPRT and a modified SPRT for speaker verification. We have demonstrated the need for robust processing techniques in making the true speaker vs. imposter decision, and have showed that the modified SPRT promises to be a good initial step in that direction. We have demonstrated on a limited speech database that the modified SPRT can achieve its design goals remarkably well, yielding an equal error rate of 5%, even allowing for variations in speaker characteristics over a prolonged time period.

9 Acknowledgements

This work has been supported by a generous grant from Ameritech Services and Bellcore.

References

[1] D. K. Burton, "Text Dependent Speaker Verification Using Vector Quantization Source Coding," *IEEE Trans. on ASSP,* vol ASSP-35, no. 2, pp. 133-143, 2/87.

[2] J. M. Naik and G. R. Doddington, "High Performance Speaker Verification Using Pricipal Spectral Components," *Proc. ICASSP,* 4/86, Tokyo, pp 881-884.

[3] Y. Tohkura, "A Weighted Cepstral Distance Measure for Speech Recognition," *Proc. ICASSP,* 4/86, Tokyo, pp 761-764.

[4] L. R. Rabiner and M. R. Sambur, "An Algorithm for Determining the Endpoints of Isolated Utterances," *Bell System Technical Journal,* vol. 54, pp. 297-315, 2/87.

[5] F. K. Soong, A. E. Rosenberg, B. H. Juang, L. R. Rabiner, "A Vector Quantization Approach to Speaker Recognition," *AT&T Technical Journal,* vol. 66, no. 2, pp. 14-36, 3/87.

[6] Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. on Comm,* vol COM-28, pp. 84-95, 1/80.

[7] H. V. Poor, *An Introduction to Signal Detection and Estimation*, Springer Verlag, New York, 1988.

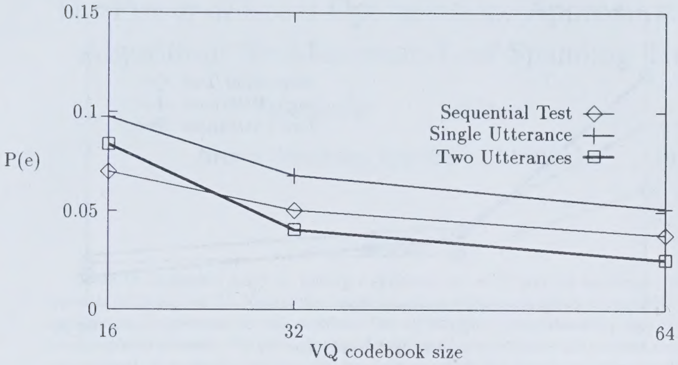


Figure 1: Average Probability of False Rejection

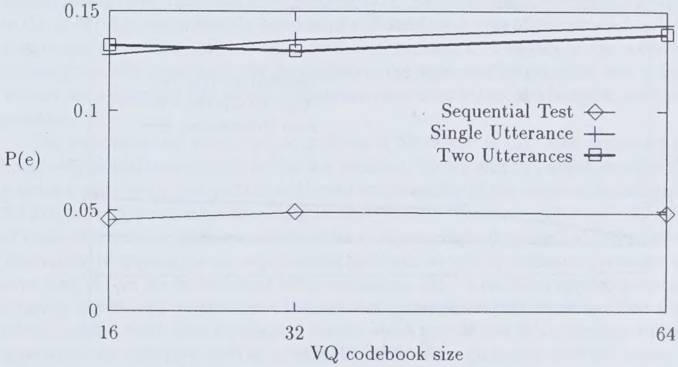


Figure 2: Average Probability of False Acceptance

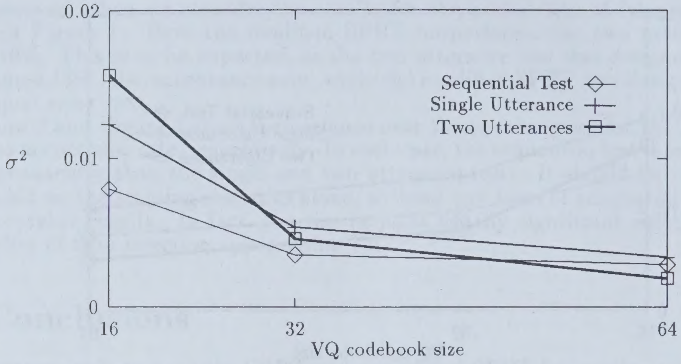


Figure 3: Error Variance of False Rejection Rate

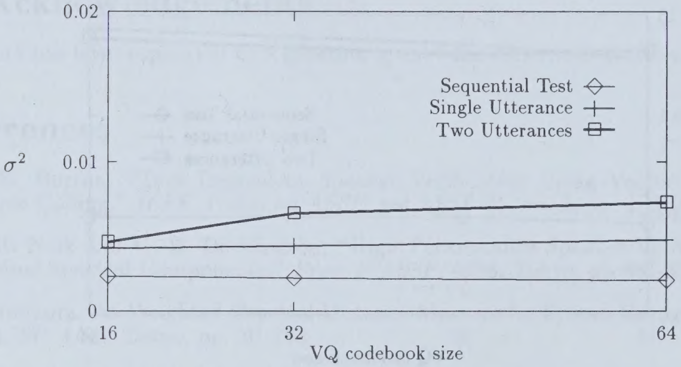


Figure 4: Error Variance of False Acceptance Rate

The Power of Local Optimization: Approximation Algorithms for Maximum-Leaf Spanning Tree

Hsueh-I Lu* R. Ravi†

Brown University, Providence, RI 02912

Abstract

Given an undirected graph G, finding a spanning tree of G with the maximum number of leaves is NP-complete [5]. We use the simple technique of local optimization to provide the first approximation algorithms for this problem. Our algorithms run in polynomial time to produce locally optimal solutions. We prove that such locally optimal solutions to this problem are globally near-optimal. In particular, we prove that two such algorithms have performance ratios of 5 and 3. The latter algorithm employs more powerful local-improvement steps than the former and hence has higher running time. This may indicate an interesting trade-off between the performance ratios and the running times of the series of algorithms we describe.

1 Introduction

Given a simple, undirected graph $G = (V, E)$, suppose we wish to find a spanning tree of G with the maximum number of leaves. This problem finds applications in communication networks, circuit layouts and in other graph-theoretic problems[17]. An interesting application is mentioned in [12]: In [3], E. W. Dijkstra studied the problem of self-stabilizing a set of processors in spite of distributed control and proposed a solution based on mutual exclusion. A variant of this solution [18] seeks a spanning tree of a graph such that the product of the degrees of all the nodes in the tree is minimum. We can use a spanning tree with the maximum number of leaves as a heuristic solution for the latter problem.

The maximum-leaf spanning tree problem is NP-complete [5]. This suggests that there might be no polynomial time algorithms for this problem. So we seek fast approximation algorithms that provide a good worst case performance ratio on the quality of the solution. In this paper, we present the first such approximation algorithms for this problem. These algorithms use the simple technique of local optimization: perform a series of local-improvement steps until a local optimum is reached. This notion of applying local-improvement heuristics to hard optimization problems was around [2] even long before the invention of NP-completeness [8]. It has been applied heuristically to solve a variety of NP-hard problems in combinatorial optimization [14]. Among all such applications of this technique, there were no positive results which proved that this technique yielded an efficient approximation algorithm with an acceptable performance guarantee until the recent work of Fürer and Raghavachari [4]. They showed that a variation of this technique applied to the minimum-degree spanning tree problem runs in polynomial time to produce near-optimal solutions. Our work is inspired by that of Fürer and Raghavachari. We demonstrate that local optimization can be used to design polynomial-time approximation algorithms for the maximum-leaf spanning tree problem with constant performance ratios.

---

*Research supported by a Graduate Fellowship from the Ministry of Education, Taiwan R. O. C. Additional support provided by NSF PYI award CCR-9157620 and DARPA contract N00014-91-J-4052 ARPA Order No. 8225.

†Research supported by an IBM Graduate Fellowship. Additional support provided by NSF PYI award CCR-9157620 and DARPA contract N00014-91-J-4052 ARPA Order No. 8225.

We propose algorithms that perform local changes that increase the number of leaves in the resulting spanning tree. For the problem at hand, there is a natural notion of such a local change, namely, swapping a tree edge for a non-tree edge. This notion can also be extended to include swapping many tree edges for an equal number of non-tree edges in a single local-change operation. Our algorithms perform such local changes that increase the number of leaves until no improvement in the solution is possible and output a locally optimal solution as the approximate solution.

By varying the limit on the number of tree edges that can participate in a single local-change operation, we derive a series of approximation algorithms. Increasing this limit corresponds to allowing for more powerful local-improvement steps. Thus, the higher is this limit, the higher is the running time of the corresponding algorithm. A local-change operation involving at most k tree edges is termed a *k-change.* Intuitively, k-changes are costlier to implement than $(k-1)$-changes. However we expect an algorithm using k-changes to yield a better solution than an algorithm using only $(k-1)$-changes. In this paper, we corroborate this intuition and prove that local-improvement algorithms that use 1-changes and 2-changes have performance ratios of 5 and 3 respectively. The following are the main results of this paper.

Theorem 1 *There is an $O(n^4)$ algorithm using 1-changes for finding a spanning tree of any undirected graph G with n nodes such that the number of leaves in this spanning tree is at least as many as a fifth of the number of leaves in any spanning tree of G.*

Theorem 2 *There is an $O(n^7)$ algorithm using 2-changes for finding a spanning tree of any undirected graph G with n nodes such that the number of leaves in this spanning tree is at least as many as a third of the number of leaves in any spanning tree of G.*

We prove that the running time of the approximation algorithm using k-changes increases with k as follows.

Lemma 1 *The running time of the approximation algorithm using k-changes on a graph with n nodes is $O(n^{3k+1})$.*

The pattern exhibited in Theorems 1 and 2 leads us to conjecture that there is a trade-off between the running times and the performance ratios exhibited in the series of algorithms we have defined.

Conjecture 1 *The performance ratio of the approximation algorithm using k-changes is strictly better than that of the algorithm using $(k-1)$-changes.*

In the next section, we discuss related work. Then we present the algorithm formally and describe the basic properties of locally optimal solutions. We then derive a rough bound on the performance guarantee of solutions optimal under 1-changes and use these ideas to prove the bound in Theorem 1. We omit the proof of the performance guarantee in Theorem 2 in this abstract. We close with suggestions for future work.

2 Related work

The idea of applying local improvements to obtain good solutions to hard optimization problems is not new [2]. It has been applied to provide heuristic solution for a variety of hard problems in combinatorial optimization. Chapter 19 of [14] examines a few applications of this technique. Some notable examples are its applications to the graph partitioning [9] and the Traveling Salesperson problems [11]. A number of complexity results relating to the paradigm of local optimality and the time complexity of computing a locally optimal solution are presented in [7, 13]. In this paper, we are interested in the application of local-search techniques to design efficient approximation algorithms, namely, those that run in *polynomial* time and provide *provably* good solutions with values close to that of the optimum.

Fürer and Raghavachari [4] showed such an application of local search to the problem of computing a spanning tree of a given graph whose maximum degree is minimum. This is termed the minimum-degree spanning tree problem and is NP-complete [5]. Fürer and Raghavachari show that local optimization can be applied to produce spanning trees and even Steiner trees of nearly minimum degree. They prove that the degree of the resulting solutions is within an additive logarithmic error of optimum. Their techniques have been generalized recently to obtain approximation algorithms for a variety of minimum-degree network design problems in [16]. In this paper, we add to the list of problems that are amenable to good approximate solutions by local-search methods.

Previous work on finding spanning trees with many leaves have focused on graphs with minimum degree at least k for some fixed $k \geq 3$. For such graphs, good lower bounds on the number of leaves achievable in a spanning tree are derived in [10, 6, 12, 17]. These lower bounds are typically proved algorithmically by constructing a spanning tree with the desired number of leaves. Thus these algorithms can be interpreted as approximation algorithms for the maximum-leaf spanning tree problem for special classes of graphs in which the minimum degree of a node is at least k. However, to the best of our knowledge, no previous approximation algorithms for this problem are known in the general case that we consider in this paper. The current best lower bound for the number of leaves achievable in a spanning tree of a n-node graph with minimum degree k is $(1 - b \ln k/k)n$ where b is any constant exceeding 2.5 and k is sufficiently large[10]. The best lower bounds for the cases $k = 3$ and 4 are $\Omega(n/4)$ and $\Omega(2n/5)$ respectively [10].

3 Definitions

In this section, we introduce some definitions that we use throughout the paper. The following definitions are relevant to any spanning tree of the input graph G. Henceforth we shall use the term degree of a node to refer to its degree in the tree under consideration. So leaves are just nodes of degree one. A node is *internal* if it is not a leaf. We call a node of degree greater than two a *high degree node*. A leaf is *special* if it is adjacent to a high degree node. All other leaves are termed *normal*. Thus normal leaves are exactly the leaves that are adjacent to nodes of degree two in the tree. A tree path containing only nodes of degree two is called a *2-path*. Its length is the number of nodes in it. A 2-path is *short* if its length is one; otherwise it is *long*. We shall refer to an edge $e = (u, v)$ as an edge *between* u and v.

For a spanning tree T, we use $\lambda(T)$ to denote the number of leaves in T. We omit the T where it is understood. We use n_i to denote the number of nodes of degree i. Suppose the input graph G has n nodes. We define $N_i = n - \sum_{1 \leq j < i} n_j$, the number of nodes with degree at least i. Thus for instance, N_3 is the number of high degree nodes. The number of short and long 2-paths are represented by P_ℓ and P_s respectively. Let λ_s and λ_n be the number of special and normal leaves respectively. We have the following observations.

Observation 1 *The number of high degree nodes in a tree is at most the number of leaves in the tree minus 2. In other words, $N_3 \leq \lambda - 2$.*

Observation 2 *The number of 2-paths in any tree is at most twice the number of leaves in the tree.*

$$P_\ell + P_s \leq N_3 + \lambda_n - 1 \leq 2\lambda - 3. \tag{1}$$

4 The Algorithm

In this section, we formally define k-changes and describe the approximation algorithms.

4.1 Edge Changes, Improvements and LOTs

Our algorithm starts with an arbitrary spanning tree T of the given graph G. Let $e = (u, v)$ be an edge in $G - T$ and f be an edge in the unique tree path of T connecting u and v. We call making e a tree edge and f a non-tree edge an *(edge) change* $(e; f)$ with respect to T. The spanning tree

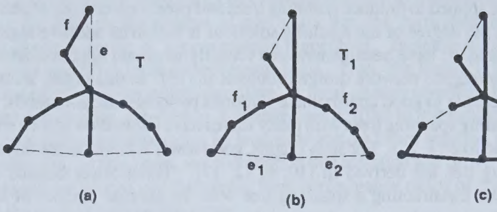


Figure 1: Examples of local improvements. In (a), the tree T is shown in dark lines and the non-tree edges are dotted. The change $(e; f)$ is a 1-improvement on T resulting in T_1 with one more leaf as shown in (b). T_1 does not admit any 1-improvement but admits the 2-improvement $(e_1, e_2; f_1, f_2)$ to give T_2 shown in (c).

obtained by applying change $(e; f)$ on T is denoted by $T(e; f)$. If $T(e; f)$ has more leaves than T, then we call the change $(e; f)$ an *improvement*.

We can generalize the notion of performing single edge swaps to allow for multiple edge swaps in a single change step. Suppose $e_0 = e$, $f_0 = f$, and $T_0 = T$. Let T_{i+1} denote $T_i(e_i; f_i)$, where $(e_i; f_i)$ is a change with respect to T_i for $0 \leq i < j$. We use $T(e_0, e_1, \ldots, e_{j-1}; f_0, f_1, \ldots, f_{j-1})$ to denote T_j. For any $k \geq j$, we define

$$(e_0, e_1, \ldots, e_{j-1}; f_0, f_1, \ldots, f_{j-1})$$

to be a k-*change* with respect to T. A k-change with respect to a spanning tree T is a k-*improvement* if the resulting spanning tree on applying the k-change has more leaves than T.

Note that a k-change is more powerful than k 1-changes. It is a batched version of any sequence of k 1-changes. In Figure 1, we illustrate a spanning tree T that admits a 1-improvement. On applying the 1-improvement we get a spanning tree T_1 that does not admit a 1-improvement but has a 2-improvement.

A k-*locally optimal tree* (k-LOT) of a given graph G is a spanning tree that does not admit any k-improvement. By definition, an i-LOT is also a j-LOT for any $j \leq i$. Since a spanning tree has exactly $n - 1$ edges, an optimal spanning tree with the maximum number of leaves is just a $(n - 1)$-LOT.

4.2 The algorithm and termination

Our idea is simply to use k-LOTs to approximate the maximum-leaf spanning tree for small values of k. We start with an arbitrary spanning tree and keep applying k-improvements on the current spanning tree until we obtain a k-LOT. Clearly, the second step dominates the time complexity. Whether a spanning tree is a k-LOT can be decided in polynomial time for constant k, since a spanning tree cannot have more than $\binom{\binom{n}{2}-n+1}{k} \times \binom{n-1}{k}$ k-changes. The algorithm itself has no more than $n - 3$ iterations because the number of leaves should be at least 2 to begin with and at most $n - 1$, and this number increases with every iteration. Therefore, the running time to obtain a k-LOT is $O(n^{3k+1})$ which is a polynomial in n for constant k. This proves Lemma 1.

Our major concern in this paper is the performance guarantee of this approximation algorithm. In other words, how much less can the number of leaves in a k-LOT be from that of the optimum? In the following sections we shall prove that the performance ratio of 1-LOTs is 5.

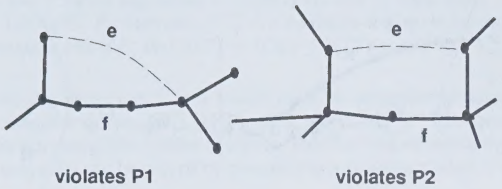


Figure 2: Illustrating properties P1 and P2 for 1-LOTs. In the above figure, dark edges represent tree edges and the dashed edge is a non-tree edge violating the property P1 or P2. In both the above cases, a 1-improvement $(e; f)$ can be identified.

5 Basic properties of 1-LOTs and a rough bound

In this section, we illustrate our proof method by proving a rough bound on the performance ratio of 1-LOTs. We begin by examining a few basic properties that are useful in proving the bounds.

5.1 Basic Properties

By definition, a 1-LOT has the following properties (See Figure 2).

P1 In a 1-LOT T, any cycle formed by adding a non-tree edge between a leaf and an internal node does not contain two adjacent nodes of degree two.

P2 In a 1-LOT T, any cycle formed by adding a non-tree edge between two internal nodes does not contain a node of degree two.

The following lemma shows that P1 and P2 together are necessary and sufficient conditions for 1-optimality.

Lemma 2 *A tree is a 1-LOT if and only if it satisfies P1 and P2.*

Proof ($\Rightarrow$) If P1 or P2 were not true for a tree T, then it is straightforward to identify a 1-improvement for T showing that T is not a 1-LOT.

($\Leftarrow$) We prove the contrapositive and show that if a spanning tree is not a 1-LOT then it violates either P1 or P2. If a spanning tree is not a 1-LOT then there exists a 1-improvement for this spanning tree. It is easy to see that any non-tree edge incident on two leaves cannot be involved in any 1-improvement. So any non-tree edge involved in a 1-improvement must be an edge between an internal node and a leaf or an edge between two internal nodes. In the first case, the tree violates property P1 and in the second case, it violates P2. ∎

Using Lemma 2, it is easy to derive the following corollaries.

Lemma 3 *In a 1-LOT T, there is no non-tree edge between a normal leaf and an internal node.*

Any 2-path in a spanning tree partitions the remaining nodes of the tree into two pieces on its removal. We refer to the sets of nodes representing these two pieces as the two *sides* of the 2-path.

Lemma 4 *In a 1-LOT T, only the first two nodes on a long 2-path, counting from a particular side, have non-tree edges to nodes in that side. Furthermore, both the endpoints of any non-tree edge between nodes in two sides of a long 2-path must be leaves.*

Lemma 5 *In a 1-LOT, there is no non-tree edge between two nodes in the same long 2-path.*

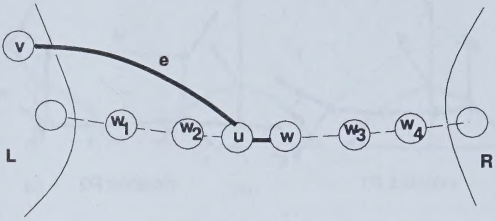


Figure 3: Proof of Lemma 6 by contradiction: The dark edges are edges of T' and the dashed edges are edges of T. If there are five leaves of T' in path $\mathcal{P}$, then the middle leaf must be connected in T' to one of the two sides via a non-tree edge e as shown above. This edge e violates the assumption that the tree T is a 1-LOT (Lemma 4).

5.2 The strategy: Proving a rough bound

In this subsection, we illustrate the basic strategy used in proving Theorem 1 by proving a weaker version of it. For this, we prove the following lemma.

Lemma 6 *Let G be an undirected graph and let T be a 1-LOT of G. Then for any spanning tree T' of G and any 2-path $\mathcal{P}$ of T, the number of leaves of T' in $\mathcal{P}$ is at most four.*

Proof The proof is by contradiction. Assume for a contradiction that there are no less than five leaves of T' in $\mathcal{P}$. Let five of them be w_1, w_2, w, w_3, and w_4, where w is the middle one along $\mathcal{P}$, and w_1 and w_2 are on the same side of w (See Figure 3). Note that by Lemma 5 there are no non-tree edges between nodes in $\mathcal{P}$. Let v be the node closest to w in T' that is not in $\mathcal{P}$. Let the last edge in the path from w to v in T' be $e = (u, v)$ where $u \in \mathcal{P}$. Since w is the middle leaf in T' along $\mathcal{P}$, the cycle closed by e contains at least two nodes of degree two, i.e., either w_1 and w_2 or w_3 and w_4. Without loss of generality, let these two nodes be w_1 and w_2. Since they are both in $\mathcal{P}$, the nodes between them in $\mathcal{P}$, if any, also have degree two in T. This implies that the cycle closed by e contains at least two adjacent nodes of degree two, which contradicts the fact that T is a 1-LOT by Lemma 4. ∎

Recall that $\lambda(T)$ denotes the number of leaves in a tree T.

Theorem 3 *Let G be an undirected graph. Let T^* be a maximum-leaf spanning tree of G and let T be a 1-LOT of G. Then $\lambda(T^*) \leq 10 \cdot \lambda(T)$.*

Proof Consider any spanning tree T' of G. We bound the number of leaves of T' that are (a) leaves, (b) high degree nodes, and (c) nodes in 2-paths in T independently. Note that these three categories account for every node in G. The total number of leaves in T is $\lambda(T)$ by definition. Also, the total number of high degree nodes in T is at most $\lambda(T)$ by Observation 1. Also, we can bound the *number* of 2-paths in T by $2\lambda(T)$ using Observation 2. Using Lemma 6, we can bound the number of leaves of T' in each 2-path of T by four. Thus the total number of leaves in T' is at most $\lambda(T) + \lambda(T) + 2\lambda(T) \cdot 4 = 10\lambda(T)$. Since this bound holds for any tree T', it also holds for T^* in particular and our theorem is proved. ∎

6 Performance Guarantee of 1-LOTs

In this section we prove the following restatement of Theorem 1.

Theorem 4 *Let G_0 be a given simple undirected graph. Let T_0^* be a maximum-leaf spanning tree of G_0 and let T_0 be a 1-LOT of G_0. Then $\lambda(T_0^*) \leq 5 \cdot \lambda(T_0)$.*

6.1 Augmentation

In order to prove Theorem 4, we augment G_0 together with T_0 as follows. For every normal leaf in T_0 adjacent to a short 2-path, we subdivide the tree edge incident on the normal leaf by inserting a new node of degree 2 in this edge. We have the following lemma.

Lemma 7 *Let G and T be the augmented versions of G_0 and T_0 respectively. If T_0 is a 1-LOT of G_0 then T is a 1-LOT of G. Furthermore, if T_0^* is a maximum-leaf spanning tree of G_0 and T^* is a maximum-leaf spanning tree of G, then $\lambda(T) = \lambda(T_0) \leq \lambda(T_0^*) \leq \lambda(T^*)$.*

Proof-sketch We can prove that T is a 1-LOT of G by contradiction using Lemma 3. Now we consider the relations among $\lambda(T)$, $\lambda(T_0)$, $\lambda(T_0^*)$, and $\lambda(T^*)$. The equality is trivial since augmentation does not change the number of leaves. The first inequality follows from the definition of T_0^*. The last inequality can be proved by constructing a spanning tree of G with at least $\lambda(T_0^*)$ leaves. ∎

6.2 Partition into regions

For the sake of proof, based on the 1-LOT T, we partition the nodes in G depending on whether they occur in a long 2-path in T. We further partition the nodes of G that are not in any long 2-path in T into *regions*. These regions are exactly the connected components of T obtained on removing the nodes in long 2-paths from T. A region is *special* if it contains at least one special leaf of T; otherwise it is *normal*. Let us use S and R to stand for the number of special and normal regions, respectively. The degree of a region is the number of long 2-paths incident on it. Note that as a result of augmentation, each normal leaf is a single-node normal region by itself. So a normal region is either a single leaf or a set of high degree nodes interconnected via short 2-paths. We term these two kinds of normal regions *external* and *internal*, respectively. It is easy to prove the following relation for a 1-LOT T. Recall that P_ℓ is the number of long 2-paths in T.

$$P_\ell = S + R - 1 \tag{2}$$

6.3 Accounting for leaves in long 2-paths

The key observation which helps proving the tighter bound in Theorem 4 is that not every long 2-path in T contributes four leaves in any spanning tree as proved in Lemma 6. The following lemma is a refined version of Lemma 6 and is proved in a similar manner.

Lemma 8 *Let T' be a spanning tree of G and $\mathcal{P}$ be a long 2-path of a 1-LOT T of G. Suppose $\mathcal{P}$ is incident on two regions $\mathcal{L}$ and $\mathcal{R}$ in T. The number of nodes in $\mathcal{P}$ which are leaves of T' is at most (i) four, if $\mathcal{L}$ and $\mathcal{R}$ are both special, (ii) three, if only one of $\mathcal{L}$ and $\mathcal{R}$ is special, and (iii) two, if $\mathcal{L}$ and $\mathcal{R}$ are both normal.*

We can think of the above lemma as providing an upper bound on the number of leaves of T' in any long 2-path $\mathcal{P}$ of T in the following way. Assume that a normal region contributes a charge of one and a special region a charge of two to the upper bound for any incident long 2-path. Then the upper bound on the number of leaves of T' in any long 2-path $\mathcal{P}$ is exactly the sum of the charge contributions of both its adjoining regions.

6.4 Invalid regions and better bounding

Even the careful counting of the leaves of T' in long 2-paths in the above lemma is not sufficient to prove Theorem 1. For example, consider an internal normal region. To connect any node in such a region in T' to any region outside, we must use one of the long 2-paths incident on it since there are no non-tree edges going from this region to any other region. This implies that this normal region cannot charge a contribution of one to *all* its incident long 2-paths. To capture this we define the notion of an *invalid* region.

Definition: A region of T is invalid in T' if it is either an internal normal region or it is a special region and every special leaf in this region is also a leaf in T'. We now define long 2-paths for which the bounds on the number of leaves of T' is more stringent than those stated in Lemma 8. Let $\mathcal{P}$ be a long 2-path of T. We call $\mathcal{P}$ *deficient* in T' if the number of nodes in $\mathcal{P}$ that are leaves in T' is at most two less than the upper bound stated in Lemma 8. We can prove the following using the properties of an invalid region of a 1-LOT.

Lemma 9 *Let k be the number of regions of a 1-LOT T that are invalid in T'. Then we can identify at least $(k-1)$ distinct long 2-paths of T that are deficient in T'.*

Now we prove our final bound on the number of leaves of any spanning tree T' that are nodes in long 2-paths in a 1-LOT T.

Lemma 10 *For any spanning tree T', the number of leaves of T' that are nodes of long 2-paths in a 1-LOT T is at most*

$$4S + 3\lambda_n - \lambda_n^* - 2S^*,$$

where λ_n^* is the number of normal leaves of T which are also leaves in T', and S^* is the number of special regions of T that are invalid in T'.

Proof Let S_i and R_i denote the number of special and normal regions of degree i for any $i \geq 1$, respectively. Then we have $S = \sum_{i\geq 1} S_i$ and $R = \sum_{i\geq 1} R_i$. If we regard T as a tree of regions, the sum of degrees of regions is at most two times the number of long 2-paths. Namely, $\sum_{i\geq 1} i \cdot S_i + \sum_{i\geq 1} i \cdot R_i \leq 2P_\ell$. We interpret Lemma 8 using the notion of charge contributions of regions to incident long 2-paths. Using this interpretation, the total number of leaves of T' in long 2-paths of T is at most $2\sum_{i\geq 1} i \cdot S_i + \sum_{i\geq 1} i \cdot R_i \leq 4P_\ell - \sum_{i\geq 1} i \cdot R_i$, by the previous inequality.

Note that a single normal leaf is also a normal region. If a normal leaf of T is also a leaf in T', then we can verify that this normal region cannot contribute any charge in the scheme above to its incident long 2-path. So we subtract one for each such region to get a bound of $4P_\ell - \sum_{i\geq 1} i \cdot R_i - \lambda_n^* = 4(R + S - 1) - \sum_{i\geq 1} i \cdot R_i - \lambda_n^*$ using (2).

Note that the number of invalid regions in T' is precisely sum of the number of internal normal regions and invalid special regions. The number of such regions is $R - R_1 + S^*$. By Lemma 9 and the definition of deficiency, we can then subtract at least $2(R - R_1 + S^* - 1)$ from the bound derived above. Thus the final upper bound on the number of leaves in T' from long 2-paths in T is

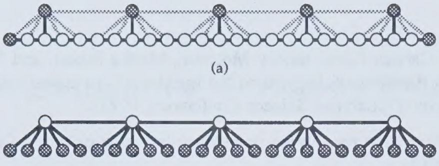
$$4(R + S) - \sum_{i\geq 1} i \cdot R_i - \lambda_n^* - 2(R - R_1 + S^*).$$

Simplifying using $R = \sum_{i\geq 1} R_i$ and $R_1 = \lambda_n$ proves the lemma. ∎

6.5 Proof of Theorem 4

We allow all high degree nodes in T and all nodes in short 2-paths of T to be leaves in T^*. Using Lemma 10 with $T' = T^*$, we infer that the number of leaves in T^* on long 2-paths of T is at most $4S + 3\lambda_n - \lambda_n^* - 2S^*$. The remaining nodes in G are just special leaves and normal leaves. Since exactly λ_n^* normal leaves are still leaves in T^*, and at least $S - S^*$ special leaves are not leaves in T', the number of leaves in T^* is bounded by

$$N_3 + P_s + (4S + 3\lambda_n - \lambda_n^* - 2S^*) + (\lambda_s - (S - S^*) + \lambda_n^*).$$


(a)

(b)

Figure 4: (a) An example of 1-LOT. (b) The maximum-leaf spanning tree of the graph in (a).

Note that $S \leq \lambda_s$ because each special region contains at least one special leaf. From (2) and (1), we get $S + P_s \leq N_3 + \lambda_n - R$. It follows that

$$\begin{aligned} \lambda(T^*) &\leq N_3 + \lambda_s + 3\lambda_n + 3S + P_s & & \\ &\leq (N_3 + 3\lambda_s + 3\lambda_n) + (N_3 + \lambda_n - R) & &\text{using } S \leq \lambda_s \\ &= (2N_3 + 3\lambda_s + 3\lambda_n) + (R_1 - R) & &\leq 2N_3 + 3\lambda_s + 3\lambda_n \\ &\leq 2n_1 + 3n_1(\text{ since } \lambda_s + \lambda_n \text{ is just } n_1) & &= 5\lambda(T). \end{aligned}$$

By Lemma 7, we know

$$\frac{\lambda(T_0^*)}{\lambda(T_0)} \leq \frac{\lambda(T^*)}{\lambda(T)} \leq 5.$$

Therefore, Theorem 4 is proved. ∎

We conclude this section by showing an example of a 1-LOT in which the bound of 5 in Theorem 1 is approachable arbitrarily closely. It is easy to verify that the tree in Figure 4-(a) is a 1-LOT. The maximum-leaf spanning tree for this graph is shown in Figure 4-(b). Clearly, the longer this 1-LOT is, the closer to five the ratio is.

7 Future work

We believe that the performance ratio of 2-LOTs is better than what we prove in Theorem 2 and conjecture the following.

Conjecture 2 *The performance guarantee of 2-LOTs is 2.5.*

Working on the cases $k \geq 3$ to obtain better performance guarantees is a direction for future work.

Papadimitriou and Yannakakis [15] identified a class of NP-hard optimization problems interreducible to one another using approximation-preserving reductions and thus took a step towards classifying NP-complete problems with respect to the hardness of approximating them. They called this class MAX-SNP. Recent work [1] has shown that problems complete for this class do not permit a fully polynomial-time approximation scheme unless $P = NP$. It is an intriguing open problem to determine if the maximum-leaf spanning tree problem is complete for MAX-SNP or if it does allow a $(1 + \epsilon)$-approximation. Along this direction, it would be interesting to examine if k-LOTs can be used in deriving such a polynomial-time approximation scheme.

Acknowledgments

We gratefully acknowledge the support, guidance and encouragement of our advisor, Prof. Philip Klein.

References

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy, "Proof Verification and Hardness of Approximation Problems," (to appear in the Thirty-third annual IEEE Foundations of Computer Science Conference, 1992).

[2] G. A. Croes, "A method for solving traveling-salesman problems," *OR 6*, No. 6, (Nov.-Dec. 1958), pp. 791-812.

[3] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Comm. ACM 17*, 11 (Nov. 1974), pp. 643-644.

[4] Martin Fürer and Balaji Raghavachari, "Approximating the minimum degree spanning tree to within one from the optimal degree", *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms* (1992), pp. 317-324.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*, W. H. Freeman, San Francisco (1979).

[6] J. R. Griggs, D. J. Kleitman, and A. Shastri, "Spanning trees with many leaves in cubic graphs, *J. Graph Theory*, 13 (1989), pp. 669-695.

[7] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?," *J. Comp. Syst. Sci.*, 37(1988), pp. 79-100.

[8] R. M. Karp, "Reducibility among combinatorial problems," in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations.* Plenum Press, New York (1972), pp. 85-103.

[9] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *BSTJ* 49 (1970), pp. 291-308.

[10] D. J. Kleitman and D. B. West," Spanning trees with many leaves," *SIAM J. Disc. Math.* Vol. 4, No. 1, (February 1991), pp. 99-106.

[11] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the Traveling-Salesman Problem," *Oper. Res.* 21 (1973), pp. 498-516.

[12] C. Payan, M. Tchuente, and N. H. Xuong, "Arbres avec un nombres maximum de sommets pendants, *Discrete Math.*, 49 (1984), pp. 267-273.

[13] C. H. Papadimitriou, A. A. Schäffer, and M. Yannakakis, "On the complexity of local search (Extended Abstract)," in *Proc. of the twenty second annual ACM Symposium on the Theory of Computing*, pp. 438-445 (1990).

[14] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc. (1982).

[15] C. H. Papadimitriou, and M. Yannakakis, "Optimization, Approximation, and Complexity classes," in *Proc. of the twentieth annual ACM Symposium on the Theory of Computing*, pp. 229-234 (1988).

[16] R. Ravi, B. Raghavachari, and P. N. Klein, "Approximation through local optimality: Designing networks with small degree," in *Proceedings, Twelfth annual conference on Foundations of Software Technology and Theoretical Computer Science* (1992), to appear.

[17] J. A. Storer, "Constructing full spanning trees for cubic graphs," *Inform. Process. Lett.* 13 (1981), pp.8-11.

[18] M. Tchuente, "Sur l'auto-stabilisation dans un réseau d'odinateurs," *R. A. I. R. O. Informatique Théorique 15*, (1) (1981), pp. 47-66.

Provably Good Algorithms for Hereditary Property Problems

Gary Chartrand[1], *Joseph McCanna*
Department of Mathematics and Statistics

Naveed Sherwani[2], *Moazzem Hossain*[3], *Jahangir Hashmi*[4]
Department of Computer Science
Western Michigan University
Kalamazoo, MI 49008

Abstract

In this paper, we propose a provably good algorithm for n-subset hereditary property problems. The class of hereditary problems is very general and includes many NP-complete problems such as maximum k-independent set problem in graphs. The proposed algorithm is simple and computes an n-subset based iteratively computing several smaller subsets. The performance ratio of the algorithm is presented for different cases. When a maximum n-subset is approximated in two steps using a maximum k-subset and a maximum $(n-k)$-subset, the performance ratio of the algorithm is $\left(1-\frac{k(n-k)}{n^2}\right)$. The performance ratio of the algorithm is $(1-(1-\frac{1}{m})^m)$, when an n-subset is computed by taking m different maximum k-subsets (*i.e.*, when n-is multiple of k). When n is not a multiple of k, the performance ratio of the algorithm is $\left(1-\frac{n-r}{n^2}\left((n-r)(1-\frac{1}{m-1})^{m-1}+r\right)\right)$, if the n-subset is computed by taking $m-1$ different k-subsets and one $(n-(m-1)k)$-subset. In its most general form, when an n-subset is computed by taking maximum k_1-, k_2-, …, k_m-subsets, for $1 \leq k_i < n$, $1 \leq i \leq m$, we show that the performance ratio of the algorithm is $(1-\Pi_{i=1}^{m}(1-r_i))$, where $r_i=\frac{k_i}{n}$.

1 Introduction

The problem of subset selection with respect to some hereditary property is a fundamental problem in graph theory, combinatorics, VLSI design automation, computational geometry, and operations research [2, 3, 5, 6, 7, 8, 9]. When stated in terms of graphs, the maximum independent set problem, the maximum clique problem, and the maximum n-colorable subgraph problem are all hereditary property based subset selection problems. Precisely stated, a graphical property Π is called *hereditary* if whenever a graph G has property Π, then so does every induced subgraph of G. Given a graph G, we denote a problem of selecting a subset of vertices in G to be *hereditary property problem* if the selected subset satisfies some hereditary property or if the selected set can be decomposed into some specified number of subsets such that each subset satisfies some given hereditary property. The problem of computing a maximum subset with respect to a hereditary

---

[1] Research supported in part by Office of Naval Research Contract N00014-91-J-1060.
[2] Research supported in part by National Science Foundation Grant USE-9052346.
[3] Research supported in part by ACM/SIGDA Scholarship 1991.
[4] The author is currently with Advanced Micro Devices, Santa Clara, CA.

property is known to be NP-hard [4]. As a result, the problem has been studied for several special cases. (For sake of clarity, we restrict our discussion to graphs; however, it is applicable to all sets.) For example, let us consider a special hereditary property subset problem known as independent set problem for graphs. Since the problem of computing a maximum independent subset of a graph is NP-hard [4], this problem has been studied for several special classes of graphs. The problem of finding a maximum n-independent set, $n \geq 1$, in a permutation graph is known to be solvable in polynomial time [3, 7]. Yannakakis and Gavril [9] considered the maximum n-colorable subgraph problem. They showed that this problem is solvable for chordal graphs in polynomial time. However, for arbitrary n, the problem remains NP-hard. For interval graphs (a special class of chordal graphs), the problem is solvable for arbitrary n in polynomial time. For circle graphs, the problem of finding a maximum independent set is solvable in polynomial time [8]; however, even for $n = 2$, the n-independent set problem is NP-hard [6].

Since the problem of finding a maximum induced subgraph with respect to a given hereditary property is NP-hard, it is unlikely that an efficient algorithm for solving such problem will be found. Consequently, for many practical applications, we need to devise some provably good approximation algorithms with guaranteed performance bounds. One approach is to develop approximation algorithms for an n-subset problem based on a maximum n'-subset problem, where $1 \leq n' < n$. If the maximum n'-subset problem with respect to a given hereditary property can be optimally computed for a given graph, then an n'-subset can be used to approximate a maximum n-subset. This reduces the problem of computing a maximum n-subset to that of computing a maximum n'-subset which is of smaller complexity problem. This approach follows the theory of continuous reduction among combinatorial optimization problems. It has been known for several years that NP-hard optimization problems differ in the degree to which their solutions can be approximated. A continuous reduction from a problem P to a problem P' has the property that if there exists an approximation algorithm for P, then there also exists an approximation algorithm for P'. Based on this approach, an $(1-(1-\frac{1}{n})^n)$-approximation algorithm has been proposed for a maximum n-independent set for circle graphs, since a maximum independent set in circle graph can be found in polynomial time [2, 5]. However, the key problem of developing provably good algorithms for an entire class of hereditary problems rather than specific instances of it has remained open.

In this paper, we consider the problem of developing provably good algorithms for n-subset hereditary property problems. Rather than develop algorithms for specific problems, we develop a general theory for the entire class of hereditary property problems. We present a simple algorithm to compute an n-subset based on some smaller subsets. The performance ratio of the algorithm is presented for different cases. When a maximum n-subset is approximated in two steps using a maximum k-subset and a maximum $(n-k)$-subset, the performance ratio of the algorithm is $\left(1 - \frac{k(n-k)}{n^2}\right)$. The performance ratio of the algorithm is $(1 - (1 - \frac{1}{m})^m)$, when an n-subset is computed by taking m different maximum k-subsets (*i.e.*, when n-is multiple of k). When n is not a multiple of k, the performance ratio of the algorithm is $\left(1 - \frac{n-r}{n^2}\left((n-r)(1 - \frac{1}{m-1})^{m-1} + r\right)\right)$, if the n-subset is computed by taking $m-1$ different k-subsets and one $(n-(m-1)k)$-subset. In its most general form, when an n-subset is computed by taking maximum k_1-, k_2-, …, k_m-subsets, for $1 \leq k_i < n$, $1 \leq i \leq m$, we formulate the problem using linear programming formulation and show that the performance ratio of the algorithm is $(1 - \Pi_{i=1}^{m}(1 - r_i))$, where $r_i = \frac{k_i}{n}$.

2 Problem Formulation and Algorithm

Let $G = (V, E)$ be a graph and let $V_1 \subseteq V$. A property Π called *hereditary property* if whenever the subgraph $\langle V_1 \rangle$ induced by V_1 has property Π, then so does the subgraph induced by V_2 for each $V_2 \subseteq V_1$, *i.e.*, for any subset $V_2 \subseteq V_1$, $\Pi(\langle V_1 \rangle) \Rightarrow \Pi(\langle V_2 \rangle$.

A set U of vertices of a graph G is called a *n-subset with respect to a hereditary property* Π ($n \geq 1$) if U can be partitioned into n subsets $U_1, U_2, \ldots, U_k$ such that the subgraph $\langle U_i \rangle$ induced by U_i has the property Π for each i ($1 \leq i \leq n$). A *maximum n-subset* (with respect to Π) is one of maximum cardinality. It is well known that for general graphs, the problem of finding a maximum n-subset with respect to any given hereditary property is NP-hard, even for $n = 1$ [4].

Despite the NP-hardness of the general problem of a maximum n-subset with respect to a hereditary property Π, the problem has many important applications in many areas including VLSI design automation, operation research, computer graphics, and operating systems. In view of all these important applications, it is necessary to develop some provably good algorithms to solve these problems. In [2], the maximum n-subset problem with respect to the hereditary property of being independent has been considered and following approximation result has been proven for circle graphs.

Theorem 1 *In a circle graph, the maximum n-independent set problem can be approximated with a lower bound of* $(1 - (1 - \frac{1}{n})^n))$ *of the maximum n-subset by taking one maximum independent set at a time.*

We consider a more generalized algorithm to find an n-subset with respect to any given hereditary property. Assume that an n-subset with respect to a hereditary property Π can be approximated by m different maximum subsets, namely by a k_1-subset, a k_2-subset, ..., a k_m-subset with respect to the same hereditary property Π. We assume that a maximum k_i-subset with respect to Π ($1 \leq i \leq m$) can be computed optimally in polynomial time.

In a graph $G = (V, E)$, the algorithm first finds a maximum k_1-subset V_1 (with respect to Π) of V, then finds a maximum k_2-subset V_2 (with respect to Π) of $V - V_1$. In the ith step, the algorithm finds a maximum k_i-subset V_i (with respect to Π) of $V - (V_1 \cup V_2 \cup \ldots \cup V_{i-1})$. Since $V_1, V_2, \ldots, V_m$ are mutually disjoint, $S' = V_1 \cup V_2 \cup \ldots \cup V_m$ is, by definition, a subset of vertices of V with respect to the same property Π. The algorithm is formally presented below:

```
Algorithm $MKS$
  Input : A graph $G = (V, E)$, positive integers $k_1$, $k_2$, ..., $k_m$,
          and a hereditary property $\Pi$;
  Output : A set of vertices representing an $n$-subset with respect
          to $\Pi$;
begin
  $V' = V$; $S = \phi$;
  for ($i = 1$ to $m$) do
      find maximum $k_i$-subset $V_i$ from $V'$ with respect to $\Pi$;
      $V' = V' - V_i$;
      $S = S \cup V_i$;
  return $S$.
end.
```

The algorithm MKS presented above approximates a maximum n-subset with respect to a given hereditary property. There are several variations that may occur depending on the values of the integers k_i's. In its simplest form, $k_1 = k_2 = \ldots = k_m = 1$ and $m = n$. Theorem 1 gives an approximation for this case for the hereditary property of being independent in a circle graph. However, the approximation holds for any hereditary property in any graph. Other cases are when the k_i are equal and $k_i \geq 1$, except possibly for k_m and when each k_i can have any value less than n. For a given approximation, if $k_i = k_j$, for $1 \leq i, j \leq m-1$, then we denote the approximation as a *monotone approximation.* Otherwise, we call the approximation *non-monotone.*

Let H_n be any heuristic algorithm. For a maximum n-subset with respect to Π in a graph G, we define the *performance ratio* of H_n to be $\frac{\mu'_n(G)}{\mu_n(G)}$, where $\mu'_n(G)$ is the size of the n-subset with respect to Π obtained by the algorithm H_n and $\mu_n(G)$ is the size of the maximum n-subset with respect to Π in G. In the subsequent sections, we present the performance ratio of the above algorithm.

3 Performance Ratio of Monotone Approximation

In this section, we present the performance ratio of a monotone approximation algorithm. There are two different cases to consider depending on the value of m, the total number of steps in the approximation algorithm. The simple case occurs when $m = 2$, *i.e.*, the approximation is performed in two steps. The other case is when $m > 2$.

3.1 Two-step Monotone Approximations

In this case the approximation is performed by first taking a maximum k_1-subset and then by taking a maximum $n - k_1$ subset from the remaining set of vertices. For ease of notation, we assume that $k_1 = k$ in the remainder of this section.

Suppose that the size of the n-subset obtained by the above algorithm is $\mu'_n(G)$ Then the performance ratio of the algorithm is

$$\rho_{n,k}(G) = \frac{\mu'_n(G)}{\mu_n(G)}. \tag{1}$$

Let $T = \bigcup_{j=1}^{k} T_j$ such that the sets $T_i \cap T_j = \phi$, for $1 \leq i, j \leq k$ and $i \neq j$. Then $\Pi(\langle T_j \rangle)$ holds for all $1 \leq j \leq k$ and $\mu_k(G) = \sum_{j=1}^{k} |T_j|$.

Since each of S_i and T_j has the same hereditary property Π, the intersection and the difference of each S_i with T_j also have the same property Π. Let $A_{ij} = S_i \cap T_j$, $B_i = S_i \backslash (\bigcup T_j)$, and $C_j = T_j \backslash (\bigcup S_i)$, for $1 \leq i \leq n$ and $1 \leq j \leq k$. See Figure 1 for an illustration. Clearly,

$$S_i = \bigcup_{j=1}^{k} A_{ij} \cup B_i, \quad \text{and} \quad T_j = \bigcup_{i=1}^{n} A_{ij} \cup C_j.$$

Now,

$$\begin{aligned} \mu_n(G) &= |S| = \sum_{i=1}^{n} |S_i| \\ &= \sum_{i=1}^{n} \sum_{j=1}^{k} |A_{ij}| + \sum_{i=1}^{n} |B_i|. \end{aligned} \tag{2}$$

Let us assume that the average cardinality of each A_{ij}, B_i, and C_j is a, b, and c, respectively. Then replacing A_{ij} and B_i in (2), we get,

$$\mu_n(G) = nka + nb.$$

	T_1	T_2	$\dots$	T_j	$\dots$	T_k	
S_1	A_{11}	A_{12}	$\dots$	A_{1j}	$\dots$	A_{1k}	B_1
S_2	A_{21}	A_{22}	$\dots$	A_{2j}	$\dots$	A_{2k}	B_2
$\vdots$							
S_i	A_{i1}	A_{i2}	$\dots$	A_{ij}	$\dots$	A_{ik}	B_i
$\vdots$							
S_n	A_{n1}	A_{n2}	$\dots$	A_{nj}	$\dots$	A_{nk}	B_n
	C_1	C_2	$\dots$	C_j	$\dots$	C_k	

Figure 1: Illustration of intersections of different subsets

Similarly,

$$\mu_k(G) = \sum_{j=1}^{k} |T_j| = \sum_{i=1}^{n}\sum_{j=1}^{k} |A_{ij}| + \sum_{j=1}^{k} |C_j| = nka + kc.$$

Since $\bigcup_{j=1}^{k} T_j$ is the maximum k-subset with respect to the hereditary property Π, any k-subset of S with respect to Π has size at most $|T|$. We define an injective selection function $\sigma_I : \{1, 2, 3, \dots, k\} \to \{1, 2, 3, \dots, n\}$, for $1 \leq I \leq \begin{pmatrix} n \\ k \end{pmatrix}$, to select a k-subset $\{S_{\sigma_I(1)}, S_{\sigma_I(2)}, \dots, S_{\sigma_I(k)}\}$ from S.

We know

$$\sum_{j=1}^{k} |T_j| \geq \sum_{i=1}^{k} |S_{\sigma_I(i)}|$$

that is

$$nka + kc \geq \sum_{i}^{k}\sum_{j}^{k} |A_{\sigma_I(i)j}| + \sum_{j}^{k} |B_{\sigma_I(j)}|.$$

Taking the summation over all possible combinations of k-subsets of S, we get

$$\begin{aligned}
\begin{pmatrix} n \\ k \end{pmatrix}(nka + kc) &\geq \sum_{I=1}^{\binom{n}{k}} \left(\sum_{i}^{k}\sum_{j}^{k} |A_{\sigma_I(i)j}| + \sum_{j}^{k} |B_j| \right), \\
\begin{pmatrix} n \\ k \end{pmatrix}(nka + kc) &\geq \begin{pmatrix} n \\ k \end{pmatrix} \left(\sum_{i}^{k}\sum_{j}^{k} a + \sum_{j}^{k} b \right) \\
nka + kc &\geq k^2 a + kb, \\
b &\leq (n-k)a + c. \qquad (3)
\end{aligned}$$

Let $k^2 a + kb = b_{\max}$. Then from (3), we get $b \leq b_{\max}$.

After computing a maximum k-subset T with respect to Π from G, the algorithm computes $U = \bigcup_{i=1}^{n-k} U_i$. By definition,

$$\mu_{n-k}(G \backslash (\bigcup_{j=1}^{k} T_j)) = \sum_{i=1}^{n-k} |U_i|.$$

The following lemma shows the relationship between the sets U_j and the sets B_i. The proof is omitted for the sake of brevity.

Lemma 1 *Let* $U = \bigcup_{i=1}^{n-k} U_i$ *be a maximum* $(n-k)$*-subset in the graph* $G\backslash(\bigcup T_j)$*. Also assume that* $\sigma : \{1, 2, 3, \ldots, n-k\} \rightarrow \{1, 2, 3, \ldots, n\}$ *is an injective function that produces a maximum (n-k)-subset from the sets* B_i *among all possible combinations of sets* B_i*. Then*

$$\sum_{i=1}^{n-k} |U_i| \;\geq\; (n-k)b,$$

where b *is the average cardinality of the sets* B_i*.*

Now, if the size of the n-subset obtained by the algorithm is $\mu'_n(G)$, then

$$\mu'_n(G) = \mu_k(G) + \mu_{n-k}(G\backslash(\bigcup T_j)) \geq nka + kc + (n-k)b = (nka + nb) + kc - kb.$$

Therefore, the performance ratio of the algorithm is

$$\begin{aligned} \frac{\mu'_n(G)}{\mu_n G} &\geq \frac{(nka + nb) + kc - kb}{nka + nb} = 1 + \frac{kc - kb}{nka + nb} \\ &\geq 1 + \frac{kc - kb_{\max}}{nka + nb_{\max}} = 1 + \frac{kc - k((n-k)a + c)}{nka + n((n-k)a + c)} \\ &= 1 - \frac{nka - k^2a}{n^2a + nc} \geq 1 - \frac{k(n-k)a}{n^2a} \\ &= 1 - \frac{k(n-k)}{n^2}. \end{aligned}$$

Thus we conclude the following:

Theorem 2 *The performance ratio of algorithm* MKS *is* $1 - \frac{k(n-k)}{n^2}$ *for a two-step monotone approximation of a maximum* n*-subset with respect to a given hereditary property* Π *by a maximum* k*-subset and a maximum* $(n-k)$*-subset with respect to* Π*.*

3.2 Multiple-step Monotone Approximation

In this section, we present the performance ratio of the monotone approximation algorithm in multiple steps. There are two cases to consider, namely, (1) $k_1 = k_2 = \ldots = k_m = k$, that is, n is a multiple of k ($n = mk$) and (2) n is not a multiple of k, that is, $k_1 = k_2 = \ldots = k_{m-1} = k$.

In case when n is multiple of k, the algorithm MKS finds an n-subset with respect to Π by determining a maximum k-subset m times from the given graph. The performance ratio of the algorithm in this case is given in the following theorem.

Theorem 3 *The performance ratio of the algorithm* MKS *is*

$$(1 - (1 - \frac{1}{m})^m)$$

for $k_1 = k_2 = \ldots = k_m = k$ *and* $n = mk$*.*

Proof: Theorem 1 was proven in [5] for $k = 1$ for the hereditary property of being independen. It is easy to see that the approximation holds for any hereditary property. Let Π'_k be the property that the vertex set of a graph can be partitioned into at most k mutually disjoint subsets, each of which satisfies the hereditary property Π. Then Π'_k is also a hereditary property. Thus, each of the maximum k-subsets with respect to Π can be considered as a set with a hereditary property Π'_k and the problem of determining a maximum n-subset with respect to Π is equivalent to finding a maximum m-subset with respect to the hereditary property Π'_k since a maximum subset with property Π'_k can be computed optimally in polynomial time. □

Let $n = (m-1)k + k_m$ for some positive integer $k \geq 1$ and $k_m \geq 0$. For ease of notation, we assume that $k_m = r$ in the remainder of this section. Let us assume that $\mu_n(G) = a$ (the size of a maximum n-subset in G with respect to Π) and $\mu_{n-r}(G) = a'$ (the size of a maximum $(n-r)$-subset in G with respect to Π). Then we have the following lemma whose proof we omit.

Lemma 2 *Given a graph G and positive integers n and r such that $n \geq r$. If a and a' are the sizes of maximum n-subset and maximum $(n-r)$-subset respectively with respect to Π, then*

$$\frac{a}{n} \leq \frac{a'}{n-r}$$

Using the above lemma, we establish the following theorem.

Theorem 4 *The performance ratio of the algorithm MKS is*

$$\left(1 - \frac{n-r}{n^2}\left((n-r)(1-\frac{1}{m-1})^{m-1} + r\right)\right)$$

for $k_1 = k_2 = \ldots = k_{m-1} = k$ and $k_m = r$, for some positive integers k and r.

Proof: Since, $n - r = (m-1)k$, a maximum $(n-r)$-subset with respect to Π can be approximated by taking $m-1$ maximum k-subsets one at a time. If the size of the maximum $(n-r)$-subset is a' and the size of the approximated $(n-r)$-subset is $\mu'_{n-r}(G)$, then by Theorem 3,

$$\mu'_{n-r}(G) \;\geq\; \left(1-(1-\frac{1}{m-1})^{m-1}\right)a'. \tag{4}$$

From lemma 2, we have

$$a' \;\geq\; \frac{a(n-r)}{n}, \tag{5}$$

where a is the size of the maximum n-subset.

Combining (4) and (5), we obtain

$$\mu'_{n-r}(G) \;\geq\; \left(1-(1-\frac{1}{m-1})^{m-1}\right)\frac{(n-r)}{n}a. \tag{6}$$

After finding an $(n-r)$-subset T_{n-r} and removing it from the graph, we see that the graph contains an n-subset of size at least

$$a - \left(1-(1-\frac{1}{m-1})^{m-1}\right)\frac{(n-r)}{n}a.$$

Therefore, we can find an r-subset of size $\mu'_r(G\backslash T_{n-r})$ such that

$$\mu'_r(G\backslash T_{n-r}) \;\geq\; \frac{r}{n}a\left[1-\frac{n-r}{n}\left(1-(1-\frac{1}{m-1})^{m-1}\right)\right]. \tag{7}$$

	V_1	V_2	.....	V_{m-1}	
S	X_1	X_2	.....	X_{m-1}	$S\backslash(\bigcup X_i)$
	Y_1	Y_2	.....	Y_{m-1}	

Figure 2: Illustration of intersection of S and V_i's

Now, if the size of an approximated n-subset is $\mu'_n(G)$, then from (6) and (7) we have

$$\begin{aligned}
\mu'_n(G) &\geq \left(1-(1-\frac{1}{m-1})^{m-1}\right)\frac{(n-r)}{n}a+\frac{r}{n}a\left[1-\frac{n-r}{n}\left(1-(1-\frac{1}{m-1})^{m-1}\right)\right] \\
&= a\left[1-\frac{n-r}{n}(1-\frac{r}{n})(1-\frac{1}{m-1})^{m-1}-\frac{r(n-r)}{n^2}\right] \\
&= a\left[1-\frac{n-r}{n^2}\left((n-r)(1-\frac{1}{m-1})^{m-1}+r\right)\right].
\end{aligned}$$

Since, $\mu_n(G)=a$, the performance ratio of this approximation algorithm is

$$\frac{\mu'_n(G)}{\mu_n(G)}=1-\frac{n-r}{n^2}\left((n-r)(1-\frac{1}{m-1})^{m-1}+r\right). \qquad \square$$

4 Performance Ratio of Non-Monotone Approximation

In this section, we consider the more generalized approximation algorithm when the value of k_i can be arbitrary and between 1 and n.

Let us assume that S is a maximum n-subset in G with respect to Π. Without loss of generality, we assume that $|S|=\mu_n(G)=1$. Then it is easy to see that $|S'|=\mu'_n(G)\leq 1$ and the performance ratio of the algorithm is

$$\frac{\mu'_n(G)}{\mu_n(G)}=\mu'_n(G).$$

We are interested in finding the worst case value of $\mu'_n(G)$.

Let $S\cap V_i=X_i$, $V_i-X_i=Y_i$, and $|X_i|=x_i$ and $|Y_i|=y_i$ for $1\leq i\leq m-1$. Then $|S\backslash(\bigcup X_i)|=1-(\sum_{i=1}^{m-1}x_i)$. Figure 2 illustrates the intersection of S and V_i for $1\leq i\leq m-1$.

Let $r_i=\frac{k_i}{n}$ for $1\leq i\leq m$. The following two lemmas show the relationships between x_i, y_i, and r_i. Their proofs are straightforward and are consequently omitted.

Lemma 3 *Let x_i and y_i be the variables defined as above and $r_i=\frac{k_i}{n}$, for $1\leq i\leq m$. Then*

$x_1+y_1\geq r_1$
$x_2+y_2\geq r_2(1-x_1)$
$\vdots$

$$x_{m-1} + y_{m-1} \geq r_{m-1}(1 - \textstyle\sum_{i=1}^{m-2} x_i)$$

Lemma 4 *Let x_i and y_i be the variables defined as above, and $r_i = \frac{k_i}{n}$, for $1 \leq i \leq m$ Then*

$$\begin{aligned} &r_2(x_1 + y_1) \geq r_1(x_2 + y_2) \\ &r_3(x_2 + y_2) \geq r_2(x_3 + y_3) \\ &\vdots \\ &r_{m-1}(x_{m-2} + y_{m-2}) \geq r_{m-2}(x_{m-1} + y_{m-1}) \end{aligned}$$

Now consider finding a maximum k_m-subset in the last step of the algorithm. After finding a maximum k_{m-1}-subset from the graph in the $(m-1)$th step, we see that the remaining graph contains an n-subset of size at least

$$1 - \sum_{i=1}^{m-1} x_i.$$

This is due to the fact that after finding a maximum k_{m-1}-subset, the size of $S \backslash (\bigcup X_i)$ is $(1 - \sum_{i=1}^{m-1} x_i)$. Therefore the size of the maximum k_m-subset is at least

$$\frac{k_m}{n}(1 - \sum_{i=1}^{m-1} x_i) = r_m(1 - \sum_{i=1}^{m-1} x_i).$$

In the worst case the size of the maximum k_m-subset is

$$r_m(1 - \sum_{i=1}^{m-1} x_i).$$

Thus in the worst case,

$$\mu'_n(G) = x_1 + y_1 + x_2 + y_2 + \ldots + x_{m-1} + y_{m-1} + r_m(1 - \sum_{i=1}^{m-1} x_i).$$

Since the objective is to find the worst case value of $\mu'_n(G)$, using Lemmas 3 and 4, we can formulate a linear programming problem with the objective of minimizing $\mu'_n(G)$ as follows:

$$\begin{aligned} &\text{minimize } z = x_1 + y_1 + x_2 + y_2 + \ldots + x_{m-1} + y_{m-1} + r_m(1 - \textstyle\sum_{i=1}^{m-1} x_i) \\ &\text{subject to,} \\ &\quad x_1 + y_1 \geq r_1 \\ &\quad x_2 + y_2 \geq r_2(1 - x_1) \\ &\quad \vdots \\ &\quad x_{m-1} + y_{m-1} \geq r_{m-1}(1 - \textstyle\sum_{i=1}^{m-2} x_i) \\ &\quad r_2(x_1 + y_1) \geq r_1(x_2 + y_2) \\ &\quad r_3(x_2 + y_2) \geq r_2(x_3 + y_3) \\ &\quad \vdots \\ &\quad r_{m-1}(x_{m-2} + y_{m-2}) \geq r_{m-2}(x_{m-1} + y_{m-1}) \\ &\quad x_1 + y_1 + x_2 + y_2 + \ldots + x_{m-1} + y_{m-1} + r_m(1 - \textstyle\sum_{i=1}^{m-1} x_i) \leq 1 \\ &\quad x_i \geq 0,\ y_i \geq 0,\ 1 \leq i \leq m-1 \end{aligned}$$

Using linear programming technique and sensitivity analysis, we show the following result. The details of the proof is omitted and may be found in [1].

Theorem 5 *The performance ratio of the non-monotone approximation of MKS algorithm is $(1 - \Pi_{i=1}^{m}(1 - r_i))$.*

5 Conclusion

In this paper, we have considered the problem of developing provably good algorithms for k-subset hereditary property problems. We have developed a general theory of approximation for the entire class of hereditary property problems. We have presented a simple algorithm to compute an k-subset based on some smaller subsets. The performance ratio of the algorithm has been presented for different cases. Our theory leads to provably good algorithms for finding an n-subset with a guaranteed performance ratio for different classes of graphs. If the maximum k-subset can be computed in polynomial time, however the complexity is very high, our algorithm can be used to approximate maximum k-subset with good performance ratio. For example, a maximum k-independent set in permutation graphs can be computed in $O(kn^2)$ time, for $k \geq 3$ and a maximum bipartite set (2-independent set) can be computed in $O(n \log n)$ time. Using a maximum bipartite set in our algorithm, a maximum k-subset can be approximated in $O(kn \log n)$ time with a performance ratio of $(1-(1-\frac{2}{k})^{k/2})$ if k is a multiple of 2; otherwise, the performance ratio is $\left(1-\frac{k-1}{k^2}\left((k-1)(1-\frac{1}{k/2-1})^{k/2-1}+1\right)\right)$.

References

[1] G. Chartrand, J. McCanna, N. Sherwani, M. Hossain, and J. Hashmi, "Provably Good Algorithms for Hereditary Property Problems", Technical report TR92-26, Dept. of Computer Science, Western Michigan University, 1992

[2] J. Cong, M. Hossain, and N. Sherwani, "A Provably Good Multilayer Topological Planar Routing Algorithm In IC Layout Design", to appear in the *IEEE Transactions on Computer-Aided Design.*

[3] J. Cong, and C. L. Liu, "On the k-Layer Planar Subset and Via Minimization Problems," *1990 European Design Automation Conference.*

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman Publishing Company, 1979.

[5] M. Hossain, and N. Sherwani, "Approximation Algorithm for Maximum k-Colorable Subgraph Problem with Application," Technical report TR91-05, Dept. of Computer Science, Western Michigan University, Kalamazoo, MI(1991).

[6] M. Sarrafzadeh and D. T. Lee, "A New Approach to Topological Via Minimization," *IEEE Trans. on CAD*, Vol. 8, No. 8, pp. 890-900, Aug. 1989.

[7] M. Sarrafzadeh, and R. D. Lou, "Maximum k-coverings of Weighted Transitive Graphs with Applications," *International Symposium on Circuits and Systems*," Singapore, 1990, pp.332-335.

[8] K. J. Supowit, "Finding a Maximum Planar Subset of a Set of Nets in a Channel," *IEEE Transactions on CAD*, Vol. CAD-6, No. 1, Jan. 1987.

[9] M. Yannakakis, and F. Gavril, "The Maximum k-Colorable Subgraph Problem for Chordal Graphs," *Information Processing Letters*, Vol. 24, pp. 133-137, Jan. 30, 1987.

Probabilistic Analysis of an Enhanced Partitioning Algorithm for the Steiner Tree Problem in R^d

Konstantinos Kalpakis and Alan T. Sherman[*]
Computer Science Department
University of Maryland Baltimore County
Baltimore, Maryland 21228-5398

September 3, 1992

Abstract

The *Geometric Steiner Minimum Tree problem (GSMT)* is to connect at minimum cost n given points (called *terminals*) in d-dimensional Euclidean space. We generalize a GSMT approximation partitioning algorithm by Komlós and Shing (KS) and analyze its performance under more relaxed conditions. These generalizations have practical applications for multi-layer VLSI routing.

Whereas KS assumed $d = 2$, our algorithm works for any dimension $d \geq 2$. Moreover, whereas their analysis assumed the rectilinear norm and a uniform distribution of input points, our analysis holds for any norm on R^d and whenever the terminals are any independent identically distributed random variables taking on values in any bounded subset of R^d.

Applying a probability theorem on subadditive Euclidean functionals by Steele, we prove the following: under the aforementioned distribution of inputs, the limit as $n \to \infty$ of the performance ratio of our algorithm is $1 + O(t^{-1/d(d-1)})$, almost surely. By adjusting the parameter t, the user can trade off time for solution quality. This result generalizes the corresponding $1 + O(t^{-1/2})$ bound proven by KS. Along the way, we prove a useful combinatorial lemma about d-dimensional rectangle slicings.

We prove that the worst-case time and space complexity of our algorithm is $\Theta(n \lg(n/t) + T_{MST}(v,v) + nT_{SMT}(t)/t)$ and $\Theta(S_{MST}(v,v) + S_{SMT}(t))$, respectively, where $v \leq n + 2^{d+1}(n/t)\sigma(t)$ is the number of vertices in the resulting Steiner tree. Here, $T_{SMT}(t)$ and $S_{SMT}(t)$ are the time and space required to solve exactly any GSMT problem of size less than t; $T_{MST}(n,m)$ and $S_{MST}(n,m)$ are the time and space required to find a minimum spanning tree of a graph with n nodes and m edges; and $\sigma(t)$ is the maximum number of Steiner points for any Steiner minimum tree with t terminals. For example, for R^2 and the rectilinear norm, the time is $O(n \lg(n/t) + n \lg^* n + nT_{SMT}(t)/t)$ and the space is $O(n \lg^* n + S_{SMT}(t))$.

Keywords. Approximation algorithms, combinatorial optimization, geometric Steiner tree problem (GSMT), graph algorithms, partitioning algorithms, probabilistic analysis of algorithms, subadditive Euclidean functionals.

---

[*] Member, Institute for Advanced Computer Studies, University of Maryland, College Park.

1 Introduction

Partitioning is a powerful heuristic that underlies many sequential and parallel approximation algorithms. We apply this heuristic to the Steiner tree problem in d-dimensional Euclidean space for any $d \geq 2$, when distances are measured by any norm on R^d. Moreover, we analyze the performance of our algorithm under a wide class of distributions of inputs by applying the theory of subadditive Euclidean functionals. This paper is of interest both for our enhanced Steiner tree algorithm and for our application of powerful general-purpose design and analysis techniques.

The Geometric Steiner Minimum Tree Problem

The *Geometric Steiner Minimum Tree problem (GSMT)* is to connect at minimum cost n given points (called *terminals*) in d-dimensional Euclidean space. Courant and Robbins [9, p. 354–361] attribute this problem to the geometer Jacob Steiner, who in the early nineteenth century, pondered the question of how to join three villages by a network of roads of minimum length; Kuhn [22] traces this problem to Fermat (1601–1665). Today, GSMT remains of interest for its applications in VLSI routing and network design [4, 33].[1]

Given any set $\hat{x}$ of n terminals in R^d with $d \geq 2$, a *Steiner Tree* $T = (V, E)$ for $\hat{x}$ is any tree that spans $\hat{x}$; *i.e.* $\hat{x} \subseteq V \subset R^d$ with $|V| < \infty$ and $E \subseteq V \times V$.[2] The points in $V - \hat{x}$ are called *Steiner points*; their use often permits lower-cost solutions than would be otherwise possible.

Given any norm $\| \ \|$ on R^d, the cost of any Steiner tree $T = (V, E)$ in R^d under $\| \ \|$ is the sum of the costs of its edges, where the *cost*—or *length*—of any edge $(x, y) \in R^d \times R^d$ is given by the induced metric $\|x - y\|$. A *Steiner minimum tree (SMT)* for $\hat{x}$ is any Steiner tree for $\hat{x}$ that has minimum cost among all Steiner trees for $\hat{x}$.

Given any dimension $d \geq 2$, any set $\hat{x}$ of n terminals in R^d, and any norm on R^d, the *Geometric Steiner Minimum Tree problem* is to compute any Steiner minimum tree for $\hat{x}$ under the given norm. The problem is geometric in the sense that the terminals and Steiner points lie in Euclidean space.

Two special cases of GSMT are the *Euclidean Steiner Minimum Tree problem (ESMT)* and the *Rectilinear Steiner Minimum Tree problem (RSMT)*. ESMT uses R^2 and the Euclidean norm $\| \ \|_2$; RSMT uses R^2 and the rectilinear norm $\| \ \|_1$.

Although *Minimum Spanning Trees (MSTs)* can be computed in polynomial time [8], GSMT is NP-hard. For example, Garey, Graham, and Johnson [14] proved that the discretized ESMT is NP-complete (reduction from Exact Cover) and that ESMT is NP-hard (it is not known if ESMT is in NP). Further, Garey and Johnson [15] proved that RSMT is NP-complete (reduction from Node Cover in planar graphs). For these reasons it would be unlikely to find a polynomial-time exact algorithm for GSMT.

There are, however, special cases in which ESMT and RSMT are efficiently solvable. For example, Aho, Garey, and Hwang [1]; Hwang [18]; and Provan [26] describe efficient algorithms for special cases of ESMT and RSMT when the terminals lie on the boundary of a convex polygon.

Previous Work

To understand our results, it is helpful to know how our work relates to previous work by Karp; Komlós and Shing; and Steele.

---

[1] Sherman's motivation for this research grew out of his experiences with the MIT PI Project [27], which uses a Steiner tree heuristic to route the signal wires globally.

[2] We exclude the case $d = 1$ because it yields the trivial Steiner minimum tree consisting of the shortest line segment containing $\hat{x}$.

In 1977, Karp [20] designed a partitioning approximation algorithm for the *Euclidean Traveling Salesman Problem (ETSP)* and claimed that his approach could be adapted to ESMT and to other geometric optimization problems.

In 1985, Komlós and Shing [21] worked out the details of a special case of Karp's claim, giving a partitioning algorithm for RSMT. Using elementary techniques, Komlós and Shing analyzed the performance ratio of their algorithm assuming the terminals are *independent identically distributed (i.i.d.)* random variables with uniform distribution. They proved the performance ratio of their algorithm is $1 + O(t^{-1/2})$ with probability approaching 1, as $n \to \infty$.

In 1981, Steele [29, 30] proved two powerful theorems about subadditive Euclidean functionals and mentioned that the cost of Euclidean and rectilinear Steiner minimum trees can be viewed as such functionals. Steele's work stems from a 1959 theorem of Beardwood, Halton, and Hammersley [3].

We combine, refine, and extend this previous work in two ways. First, we generalize the algorithm of Komlós and Shing to work in d-dimensional Euclidean space, for any $d \geq 2$. Second, by applying Steele's theorems, we analyze the performance ratio of Algorithm 1 under more general conditions. Namely, we prove the following: for any norm on R^d and whenever the terminals are any i.i.d. random variables taking on values in any bounded subset of R^d, the limit as $n \to \infty$ of the performance ratio of Algorithm 1 is $1 + O(t^{-1/d(d-1)})$, almost surely. Thus, we generalize the algorithm of Komlós and Shing, improve its analysis, and work out the details of claims by Karp and Steele.

Although the generalization to R^d with $d > 3$ is primarily of theoretical interest, the generalization to R^3 has important practical applications. For example, R^3 provides a useful model for multi-layer VLSI routing. In addition, by relaxing the constraints on the norm and on the input distribution, we can apply our results to VLSI routing problems with non-rectilinear wires and to more realistic input distributions.

Komlós and Shing stated the time complexity of their algorithm as $O(n \lg n + nT_{SMT}(t))$, where $T_{SMT}(t)$ is the time to solve one subproblem of size t; they did not state the space complexity of their algorithm. By contrast, for our algorithm under their assumptions (*i.e.* R^2 and the rectilinear norm), we give the tighter bounds of $O(n \lg(n/t) + n \lg^* n + nT_{SMT}(t)/t)$ time and $O(n \lg^* n + S_{SMT}(t))$ space.[3]

Outline

The rest of this paper is organized as follows. In Section 2, we describe our algorithm in detail, prove its correctness, and analyze its time and space complexity. In Section 3 we briefly review Steele's work on subadditive Euclidean functionals and apply this work to the asymptotic cost of any Steiner minimum tree in R^d using any norm on R^d. In Section 4, we prove a combinatorial lemma about rectangle slicings and apply this lemma, together with the results from Section 3, to analyze the asymptotic performance ratio of Algorithm 1. Finally, in Section 5, we summarize our conclusions.

2 Algorithm 1

In this section we describe our Steiner tree algorithm, prove its correctness, and analyze its time and space complexity. We call this approximation algorithm *Algorithm 1.*

2.1 Description of Algorithm 1

Given any set of n terminals in R^d, Algorithm 1 computes an approximate Steiner minimum tree in three steps. First, it partitions the terminals into $\Theta(n/t)$ sets, each of size at most t. Second, using an exact GSMT component algorithm, Algorithm 1 constructs a Steiner minimum tree for each of these sets. Third, Algorithm 1 constructs and outputs a minimum spanning tree for the union of the Steiner minimum trees found in Step 2. The parameter $2 \leq t \leq n$ controls the maximum size of each subproblem; by adjusting this parameter, the user can conveniently trade off time for solution quality.

---

[3] Although Komlós and Shing did not state the entire effect of t on the running time of their algorithm, from their calculations, it is apparent they were aware of its effect.

Algorithm 1 recursively partitions the terminals as follows: at each step, Procedure Partition divides the current set into 2^d subsets, using the medians along each of the d coordinates. Procedure Split_Set partitions any set of terminals along one dimension. For details, see Kalpakis and Sherman [19]. Without loss of generality, we assume all terminals have unique coordinates along each dimension.

For the exact GSMT component algorithm, a variety of options are available. For example, Smith [28] gives a method for finding Steiner minimum trees in R^d, for any $d \geq 2$ under the Euclidean norm. For R^2 under the Euclidean norm, there are algorithms by Melzak [23], Cockayne and Schiller [7], Boyce [5], Winter [32], Trietsch and Hwang [31], and Cockayne and Hewgill [6]. For R^2 and the rectilinear norm, by Hanan's theorem [17], we can use any Steiner minimum tree algorithm for graphs; Winter [32] reviews such algorithms. Although our performance analysis is based on using an exact GSMT component algorithm, the user may wish to carry out additional time-quality tradeoffs by using an approximate component algorithm.

To compute the minimum spanning tree, we use the algorithm by Fredman and Tarjan [13], which runs in $O(E \lg^* V)$ time, where $|E| \geq |V|$ are, respectively, the number of edges and vertices.[4] Although unimportant to our theoretical asymptotic bounds, in practice we recommend refining the output of Algorithm 1 with post-processing heuristics surveyed by Balakrishnan and Patel [2].

We note that Algorithm 1 is correct in the sense that it always terminates and, upon termination, finds some Steiner tree for the input terminals. To prove that the output graph is a Steiner tree, it suffices to prove that the union of the Steiner trees for the subproblems is connected and spans all terminals. The connectness property holds because Procedure Split_Set includes the median-coordinate terminal in both subsets of the partition; the spanning property holds because every terminal is in some subproblem.

2.2 Time and Space Complexity

We analyze the time and space complexity of Algorithm 1 under the uniform cost measure. In Proposition 1, we state the complexity of Algorithm 1 in terms of the parameter t and the complexities of the component algorithms that solve the GSMT and MST subproblems. Corollary 1 specializes this lemma for R^d using the Euclidean norm and for R^2 using the rectilinear norm.

The complexity of Algorithm 1 depends on the following functions. Let $T_{SMT}(n)$ and $S_{SMT}(n)$ be the time and space complexity of any exact component GSMT algorithm, where n is the number of terminals. Similarly, let $T_{MST}(V, E)$ and $S_{MST}(V, E)$ be the time and space complexity of any MST algorithm on $|V|$ vertices and $|E|$ edges. Also, let $\sigma(n)$ denote the maximum number of Steiner points for any Steiner minimum tree for n terminal points. We assume that $T_{SMT}(n)$, $S_{SMT}(n)$, $\sigma(n)$, $T_{MST}(n, m)$, and $S_{MST}(n, m)$ are asymptotically non-decreasing functions of n and m.

It is helpful to visualize the partition tree of height $\lceil (1/d)\lg(n/(t-1)) \rceil$ formed by the partitioning process. Each node of this tree represents a set of terminals, with the z leaves being the GSMT subproblems to be solved exactly, where $n/(t-1) \leq z \leq 2^d(n/(t-1)) \leq 2^{d+1}(n/t)$. Each internal node has exactly 2^d children; for each level k, each of the 2^{dk} nodes at level k corresponds to at least $\lfloor n/2^{dk} \rfloor$ and to at most $\lfloor n/2^{dk} \rfloor + 1$ terminals.

---

[4] Throughout this paper, inside asymptotic notation and only inside asymptotic notation, if S is any set, we use the shorthand S to denote $|S|$. For example, $O(S)$ denotes $O(|S|)$.

Proposition 1. *For any $d \geq 2$, for R^d under any norm, the time and space complexity of Algorithm 1 is $\Theta(n \lg(n/t) + T_{MST}(v, v) + nT_{SMT}(t)/t)$ and $\Theta(S_{SMT}(t) + S_{MST}(v, v))$, respectively, where n is the number of terminals and $v \leq n + 2^{d+1}(n/t)\sigma(t)$ is the number of vertices in the resulting Steiner tree.*

Proof. The running time of Algorithm 1 is the sum of four components: $\Theta(n \lg(n/t))$ time to partition the terminals, $\Theta((n/t)T_{SMT}(t))$ time to solve the z subproblems, $\Theta(v)$ time to compute the union the Steiner trees from the subproblems (this forest has $v \leq n + z\sigma(t)$ terminals and Steiner points), and $T_{MST}(v, v) \in \Omega(v)$ time to compute a minimum spanning tree of this forest. Note that the median of n real numbers can be computed in $\Theta(n)$ time and that each of the operations Insert, Delete, Empty, and Append takes time $\Theta(1)$.

The $\Theta(n \lg(n/t))$ time to partition the terminals satisfies the recurrence

$$A(n) = \begin{cases} n & \text{if } n \leq t \\ 2^d A(n/2^d) + \Theta(dn) & \text{if } n > t, \end{cases} \tag{1}$$

where the $\Theta(dn)$ term reflects that each call to Procedure Partition(S, d) takes time $\Theta(dS)$.

Similarly, the space complexity of Algorithm 1 is the sum of four terms: $\Theta(n)$ space to partition the terminals, $S_{SMT}(t)$ space to solve the subproblems (we reuse space), $\Theta(v)$ space to compute the union the Steiner trees from the subproblems, and $S_{MST}(v, v) \in \Omega(v)$ space to compute a minimum spanning tree. □

Corollary 1. *With R^d under the $\| \|_2$ norm for any $d \geq 2$, or with R^2 under the $\| \|_1$ norm, the time and space complexity of Algorithm 1 is $O(n \lg(n/t) + n \lg^* n + nT_{SMT}(t)/t)$ and $O(n \lg^* n + S_{SMT}(t))$, respectively, where n is the number of terminals.*

Proof. For R^d under the $\| \|_2$ norm, Gilbert and Pollak [16] proved that $\sigma(n) \leq n - 2$; for R^2 under the $\| \|_1$ norm, Komlós and Shing [21] proved that $\sigma(n) \leq 2n$. Further, Fredman and Tarjan [13] give an MST algorithm based on Fibonacci heaps that runs in time $O(E \lg^* V)$ on any connected graph with $|V|$ vertices and $|E|$ edges. Hence, the corollary follows from Lemma 1 with $v \leq n + z\sigma(t) \leq n + 2^{d+1}(n/t)2t \leq (2^{d+2} + 1)n$ and $S_{MST}(v) \leq T_{MST}(v) \in O(v \lg^* v)$. □

3 Asymptotic Cost of Geometric Steiner Minimum Trees

To evaluate the performance of Algorithm 1, we need to estimate the cost of any Steiner minimum tree for any set of n terminals in R^d using the chosen norm. Since the cost of such Steiner minimum trees can be viewed as a subadditive Euclidean functional, we can estimate this cost by applying a theorem by Steele [29]. In this section, we review and apply this theorem.

To this end, given any finite set $\hat{x}$ of points in R^d, let $L_S(\hat{x})$ be the cost of any Steiner minimum tree for $\hat{x}$ using the chosen norm. Similarly, let $\delta_2(\hat{x})$ be the length of any shortest path through these points using the Euclidean norm.

3.1 Subadditive Euclidean Functionals

We shall use the following terminology throughout.

A *functional* is any function $L : \mathcal{P}(R^d) \rightarrow R$, where $\mathcal{P}(R^d)$ is the set of all finite subsets of R^d. The functional L is *Euclidean* iff *L dilates linearly* and L is *invariant under translation*; that is, for any $\hat{x} \in \mathcal{P}(R^d)$ and for any $\zeta \in R$, $L(\zeta \odot \hat{x}) = |\zeta| L(\hat{x})$ and $L(\zeta \oplus \hat{x}) = L(\hat{x})$. The *dilation* operator $\odot : R \times R^d \rightarrow R^d$ and the *translation* operator $\oplus : R \times R^d \rightarrow R^d$ are defined as follows. For any point $y = (y_1, y_2, \ldots, y_d) \in R^d$ and any constant $\zeta \in R$, $\zeta \odot y = (\zeta y_1, \zeta y_2, \ldots, \zeta y_d)$ and $\zeta \oplus y = (\zeta + y_1, \zeta + y_2, \ldots, \zeta + y_d)$. We lift these operators to act on sets of points as follows: $\zeta \odot \hat{x} = \{\zeta \odot x_1, \zeta \odot x_2, \ldots, \zeta \odot x_n\}$, and $\zeta \oplus \hat{x} = \{\zeta \oplus x_1, \zeta \oplus x_2, \ldots, \zeta \oplus x_n\}$, whenever $\hat{x} = \{x_1, x_2, \ldots, x_n\}$ is any set of points in R^d. For convenience, we will sometimes write $\zeta\hat{x} = \zeta \odot \hat{x}$.

A Euclidean functional $L : \mathcal{P}(R^d) \to R$ is *monotone* iff, for any $\hat{x} \in \mathcal{P}(R^d)$ and any $y \in R^d$, $L(\hat{x}) \leq L(\hat{x} \cup \{y\})$. And L is *subadditive* iff L satisfies Steele's subadditive hypothesis: there exists some positive real constant $C_L(d)$ such that, for any set $\hat{x}$ of n terminals in R^d, for any positive integer m, for any positive $\zeta \in R$, and for any partition $\{Q_i : i = 1, \ldots, m^d\}$ of the unit cube $[0,1]^d$ into m^d identical subcubes with edges parallel to the axes, $L(\hat{x} \cap [0,\zeta]^d) \leq C_L(d)\zeta m^{d-1} + \sum_{i=1}^{m^d} L(\hat{x} \cap \zeta Q_i)$. Given any real numbers $a < b$, the *d-dimensional cube* $[a,b]^d$ is the set $\{x : x = (x_1, \ldots, x_d) \in R^d \text{ and } a \leq x_i \leq b\}$.

Next, we review a powerful theorem by Steele [29] concerning the asymptotic value of subadditive Euclidean functionals. For definitions of scale bounded, simply subadditive, and upper linear, see Steele [29].

Theorem 1 (Steele). *Let n and $d \geq 2$ be any positive integers. Let $\hat{X}$ be any set of n i.i.d. random variables in R^d with probability distribution of bounded support and absolutely continuous part $f(x)$. Let $L : \mathcal{P}(R^d) \to R$ be any monotone, scale bounded, simply subadditive, upper linear, subadditive Euclidean functional that has bounded variance. Then there exists some positive real constant $\beta_L(d)$ such that*

$$\lim_{n\to\infty} \frac{L(\hat{X})}{n^{(d-1)/d}} = \beta_L(d) \int_{R^d} f(x)^{\frac{d-1}{d}}\, dx \tag{2}$$

almost surely.[5]

Steele uses the properties of scale boundedness and simple subadditivity to show that, if any set of n i.i.d. random variables $\hat{X}$ is restricted to the singular support of their common *probability density function (p.d.f.)*, then their contribution to L is small: by Lemma 3.1 in Steele [29], their contribution is only $o(n^{(d-1)/d})$ almost surely.[6]

Finally, we review a theorem by Few [12] that we use to bound the cost of trees computed by Algorithm 1. This theorem bounds the cost of shortest paths through points in R^d and can be viewed as a special case of Theorem 1.

Theorem 2 (Few). *Let $d \geq 2$ be any positive integer. There exists some positive real constant c_d such that, for any positive real ζ and for any set of n points $\hat{x} \in [0,\zeta]^d$, $\delta_2(\hat{x}) \leq c_d\zeta\, n^{(d-1)/d}$.*

3.2 Application of Steele's Theorem

We apply Steele's theorem to find the asymptotic cost of geometric Steiner minimum trees in R^d when the norm is arbitrary and when the terminals are any set $\hat{X}$ of n i.i.d. random variables taking on values in any bounded subset of R^d. For this case, we prove the cost of any Steiner minimum tree for $\hat{X}$ converges as $n \to \infty$ to $\Theta(n^{(d-1)/d})$ almost surely. This result is similar in spirit to an analogous but less general result proven by Karp [20] for ETSP.

A norm $\| \|$ on R^d is *Euclidean bounded* iff there exists some positive real constant ξ such that, for any $x \in R^d$, $\|x\| \leq \xi\|x\|_2$. We use this property in conjunction with Theorem 2 to derive various upper bounds. Note that every norm on R^d satisfies this property. For convenience, we assume the terminals are chosen from some bounded subset $D \subset R^d$, which implies that the p.d.f. of the terminals has bounded support.

To apply Theorem 1, we must prove that for any norm on R^d, the function $L_S : \mathcal{P}(D) \to R$ satisfies the hypotheses of Steele's theorems; we do so in Lemma 1.

---

[5] All integrals in this paper are Lebesgue integrals with respect to the d-dimensional Lebesgue measure [11].

[6] The *support* of any function $g : R^d \to R$ is the set $\{x \in R^d : g(x) \neq 0\}$. By the Lebesgue Decomposition Theorem [24, p. 216] any p.d.f. can be written as the sum of two functions: one that is is absolutely continuous and one that is singular with respect to Lebesgue measure. The support of the singular part of any distribution is called the *singular support.*

Lemma 1. *For any dimension $d \geq 2$ and for any norm on R^d, the function L_S is a monotone, scale bounded, simply subadditive, upper linear, subadditive Euclidean functional. Furthermore, if $\hat{X}$ is any set of n i.i.d. random variables taking on values in D, then $L_S(\hat{X})$ has bounded variance.*

Proof. Most of the properties are straightforward albeit tedious to prove; subadditivity and upper-linearity require the most work. For details, see Kalpakis and Sherman [19]. □

Proposition 2. *Let $d \geq 2$ be any positive integers and let $\hat{X}$ be any set of n i.i.d. random variables taking on values in any bounded subset of R^d with any distribution of absolutely continuous part $f(x)$. For any norm on R^d, there exist positive real constants $\beta_{L_S}(d)$ and $\beta'_{L_S}(d)$ such that*

$$\lim_{n\to\infty} \frac{L_S(\hat{X})}{n^{(d-1)/d}} = \beta_{L_S}(d) \int_{R^d} f(x)^{(d-1)/d} dx \tag{3}$$

almost surely, and

$$\lim_{n\to\infty} E\left(\frac{L_S(\hat{X})}{n^{(d-1)/d}}\right) = \beta'_{L_S}(d) \int_{R^d} f(x)^{(d-1)/d} dx. \tag{4}$$

Proof (direct proof). Equation 3 follows immediately from Lemma 1 and Theorem 1. To prove Equation 4, we begin with Equation 3 and apply the Dominated Convergence Theorem (Moran [24, p. 204]). To this end, let $[0, \zeta]^d$ be the smallest d-cube that contains D, the bounded set from which the terminals are drawn. To apply the Dominated Convergence Theorem, it suffices to show there exists some positive real constant $B_{L_S}(d)$ such that $L_S(\hat{X})/n^{(d-1)/d} \leq B_{L_S}(d)\zeta$, which follows from the scale boundedness of L_S. □

Note that when the terminals are i.i.d. random variables uniformly distributed over $[0,1]^d$, the integrals in Equations 3 and 4 equal 1.

4 Probabilistic Performance Analysis of Algorithm 1

We evaluate Algorithm 1 in terms of its *performance ratio* $R_A(\hat{X}) = c(T_{APR}(\hat{X}))/c(T_{OPT}(\hat{X}))$, where $T_{APR}(\hat{X})$ is the Steiner tree found by Algorithm 1 for terminals $\hat{X}$; $T_{OPT}(\hat{X})$ is any Steiner minimum tree for $\hat{X}$; and c is the cost function for Steiner trees under the chosen norm. First, we compute upper bounds on $c(T_{APR}(\hat{X}))$, making use of a novel combinatorial lemma about the sum of perimeters of rectangle slicings. Second, we combine these bounds with the asymptotic cost of $T_{OPT}(\hat{X})$ computed in Section 3 to obtain an asymptotic upper bound on the performance ratio.

4.1 Slicings of Rectangles in Euclidean Space

Let $\hat{x}$ be any finite set of terminals contained in any d-rectangle $R_0 \subset R^d$. As Algorithm 1 partitions $\hat{x}$, there is a natural corresponding slicing of R_0. To bound the cost of the Steiner tree computed by Algorithm 1, it is helpful to know the sum of perimeters of the rectangles in this slicing. Lemma 2 computes this sum. Before presenting this lemma, we define slicing and some related concepts.

Let $Q = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_d, b_d]$ be any d-rectangle in R^d for any $d \geq 2$, and let $1 \leq j \leq d$. A *slicing of Q along coordinate j* is any partition $\{Q_L, Q_R\}$ of Q into any two d-rectangles Q_L, Q_R such that $Q_L = [a_1, b_1] \times \ldots \times [a_{j-1}, b_{j-1}] \times [a_j, c_j] \times [a_{j+1}, b_{j+1}] \times \ldots \times [a_d, b_d]$ and $Q_R = [a_1, b_1] \times \ldots \times [a_{j-1}, b_{j-1}] \times [c_j, b_j] \times [a_{j+1}, b_{j+1}] \times \ldots \times [a_d, b_d]$, for some real number $c_j \in (a_j, b_j)$. We say that the slicing is *uniform* iff $c_j = (a_j + b_j)/2$.

Since Procedure 1 partitions the terminals along each of the d dimensions, it is helpful to introduce the following more general notion of slicing. Let $W \subseteq \{1, 2, \ldots, d\}$ be any set of coordinates. *A slicing of Q along the set of coordinates W* is defined recursively as follows: if $W = \emptyset$, it is $\{Q\}$; otherwise, it is $S_1 \cup S_2$, where S_1 and S_2 are the slicings of Q_1 and Q_2 along the set of coordinates $W - \{i\}$, respectively, where $\{Q_1, Q_2\}$ is any slicing of Q along any coordinate $i \in W$.[7] This slicing is *uniform* iff each of the slicings S_1, S_2, and $\{Q_1, Q_2\}$ is uniform.

Procedure 1 partitions terminals $\hat{x}$ into sets $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{2^d}$. A *corresponding slicing* of the bounding rectangle R_0 is any slicing $\{Q_1, Q_2, \ldots, Q_{2^d}\}$ of R_0 along the set of coordinates $\{1, 2, \ldots, d\}$ such that, for any integer $1 \leq i \leq 2^d$, $\hat{x}_i \subseteq Q_i$.

The following lemma computes the sum of the perimeters of the rectangles in any slicing of Q along the set of coordinates $\{1, 2, \ldots, d\}$. To this end, let $\Gamma(Q)$ denote the *perimeter* of rectangle Q, measured by the Euclidean norm.

Lemma 2 (Slicing Lemma). *Let $d \geq 1$ be any integer and let Q be any rectangle in R^d. If $\{Q_1, Q_2, \ldots, Q_{2^d}\}$ is any slicing of Q along the set of coordinates $\{1, 2, \ldots, d\}$, then $\sum_{j=1}^{2^d} \Gamma(Q_j) = 2^{d-1}\Gamma(Q)$.*

Proof (by counting). We prove the lemma assuming uniform slicings, since for any slicing $\{Q_L, Q_R\}$ of any rectangle Q along any coordinate $1 \leq i \leq d$, $\Gamma(Q_L) + \Gamma(Q_R)$ depends only on i and Q. For details, see Kalpakis and Sherman [19]. □

4.2 An Upper Bound on the Cost of the Steiner Tree Computed by Algorithm 1

To compute an upper bound on the performance ratio of Algorithm 1, we need an upper bound on the cost of any Steiner tree computed by Algorithm 1. In Lemma 3, we prove such a bound from our combinatorial lemma.

Lemma 3. *Let $d \geq 2$ be any dimension; let $\hat{x}$ be any set of n terminals in R^d; and let R_0 be any d-rectangle that contains $\hat{x}$. For any norm on R^d, there exist positive real constants a_1 and a_2, such that*

$$c(T_{APR}(\hat{x})) \leq c(T_{OPT}(\hat{x})) + a_1\Gamma(R_0)\left(\frac{n}{t}\right)^{\frac{d-1}{d}} + a_2\Gamma(R_0)t^{(d-2)/(d-1)}\left(\frac{n}{t}\right)^{\frac{d-1}{d}}, \tag{5}$$

where $t \geq 1$ is the parameter in Algorithm 1.

Proof (by construction). As Algorithm 1 partitions the terminals $\hat{x}$ into subsets $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{2^{\mu d}}$, it creates a corresponding slicing $R_1, R_2, \ldots, R_{2^{\mu d}}$ of R_0 along coordinates $\{1, 2, \ldots, d\}$, where $\mu = \lceil (1/d)\lg(n/(t-1)) \rceil$. Thus, each rectangle R_i contains the set $\hat{x}_i$. For each $1 \leq i \leq 2^{\mu d}$, the component algorithm computes a Steiner minimum tree T_i for $\hat{x}_i$. The Steiner tree computed by Algorithm 1 is a subtree of the connected graph formed by the union of these T_i's.

We bound the cost of $T_{APR}(\hat{x})$ in three steps: First, for each i, we construct a connected graph G_i spanning $\hat{x}_i$ from the restriction of $T_{OPT}(\hat{x})$ in R_i. Second, using Theorem 2, we find an upper bound on $c(G_i)$ for each G_i. Third, applying our slicing lemma, we find an upper bound on $\sum_{i=1}^{2^{\mu d}} c(G_i)$, which is an upper bound on $T_{APR}(\hat{x})$ since $c(T_i) \leq c(G_i)$ for all $1 \leq i \leq 2^{\mu d}$.

Step I: Constructing G_i.

---

[7] Note that a slicing along a set of coordinates depends on the choice of i at each step; Algorithm 1 always chooses $i = \min(W)$.

For each $1 \leq i \leq 2^{\mu d}$, we construct a connected graph G_i spanning $\hat{x}_i$ from the restriction $G'_i = T_{OPT}(\hat{x}) \cap R_i$. If G'_i is connected, then choose $G_i = G'_i$; otherwise, connect G'_i by adding some extra points and edges as follows.

For each connected component Z_k of G'_i, Z_k has some edge that intersects some facet $F_{i,j}$ of R_i at some point $a_{i,j,k}$.[8] Let $A_{i,j} = \{a_{i,j,k} : 1 \leq k \leq cc(i)\}$, where $cc(i)$ is the number of connected components of G'_i. For each facet $F_{i,j}$ of R_0, add to G'_i the points in $A_{i,j}$ and the edges along any shortest path through these points.

Further augment G'_i as follows. From each non-empty set $A_{i,j}$, select one point $v_{i,j}$; let V_i be the collection of these points. Add to G'_i the points in V_i and the edges along any shortest path through them. The resulting graph G_i is connected and spans $\hat{x}_i$.

Step II: Computing an upper bound on the cost of G_i.

By the construction of G_i, it follows that

$$c(G_i) \leq c(T_{OPT}(\hat{x}) \cap R_i) + \delta(V_i) + \sum_{j=1}^{f_d} \delta(A_{i,j}), \tag{6}$$

where $f_d = 2d$ is the number of facets in any d-rectangle, and $\delta(\cdot)$ denotes the cost—under the specified norm—of any shortest path through any set of points in R^d.

We now separately bound $\delta(V_i)$ and $\delta(A_{i,j})$. Since V_i contains at most f_d points, $\delta(V_i) \leq \xi f_d \Gamma(R_i)$, where ξ is the constant from the Euclidean-boundedness property of the specified norm. Next, consider any $1 \leq j \leq f_d$. Since $A_{i,j}$ is contained in facet $F_{i,j}$, which is a $(d-1)$-rectangle, we can apply Theorem 2 with $\zeta = e_{i,j}$, where $e_{i,j}$ is the maximum cost of any edge of $F_{i,j}$ under $\| \, \|_2$. By Theorem 2, there exists some constant c_{d-1} such that $\delta(A_{i,j}) \leq \xi c_{d-1} e_{i,j} |A_{i,j}|^{(d-2)/(d-1)}$.

Using the facts that $e_{i,j} \leq \Gamma(R_i)/2^{d-1}$ and $|A_{i,j}| \leq |\hat{x}_i| \leq t$, Inequality 6 yields

$$c(G_i) \leq c(T_{OPT}(\hat{x}) \cap R_i) + \xi f_d \Gamma(R_i) + \xi c_{d-1} f_d 2^{-(d-1)} \Gamma(R_i) t^{(d-2)/(d-1)}. \tag{7}$$

Step III: Computing an upper bound on $\sum_{i=1}^{2^{\mu d}} c(G_i)$.

Recursively applying Lemma 2, we have $\sum_{i=1}^{2^{\mu d}} \Gamma(R_i) = 2^{\mu(d-1)} \Gamma(R_0)$. Therefore, from Inequality 7, it follows that

$$\sum_{i=1}^{2^{\mu d}} c(G_i) \leq c(T_{OPT}(\hat{x})) + \xi f_d 2^{\mu(d-1)} \Gamma(R_0) + \xi c_{d-1} f_d 2^{-(d-1)} t^{(d-2)/(d-1)} 2^{\mu(d-1)} \Gamma(R_0). \tag{8}$$

Because $c(T_{APR}(\hat{x})) \leq \sum_{i=1}^{2^{\mu d}} c(G_i)$, the lemma follows from Inequality 8 and the fact that $2^{\mu d} \in \Theta(n/t)$. □

4.3 Upper Bounds on the Performance Ratio of Algorithm 1

To find an asymptotic upper bound on $R_A(\hat{X})$, we combine Proposition 2 and Lemma 3.

Theorem 3. *Let $d \geq 2$ be any positive integer and let $\hat{X}$ be any set of n i.i.d. random variables taking on values in any bounded subset of R^d with any p.d.f. of absolutely continuous part $f(x)$. Then,*

$$\limsup_{n \to \infty} R_A(\hat{X}) = 1 + O\left(\frac{t^{-1/(d(d-1))}}{\int_{R^d} f(x)^{(d-1)/d} dx} \right) \tag{9}$$

almost surely, and

---

[8] Each facet of R_0 is a $(d-1)$-rectangle [25].

$$\limsup_{n\to\infty} E(R_A(\hat{X})) = 1 + O\left(\frac{t^{-1/(d(d-1))}}{\int_{R^d} f(x)^{(d-1)/d}dx}\right), \tag{10}$$

where $t \geq 1$ is the parameter in Algorithm 1.

Proof (direct proof). To prove Equation 9, we divide both sides of Inequality 5 by $c(T_{OPT}(\hat{X})) > 0$, take limits, and apply Equation 3 of Proposition 2. After some algebra and because $d > 1$, the dominant non-constant term on the right-hand side of the resulting inequality is $t^{-1/(d(d-1))}$. The proof of Equation 10 is similar but uses Equation 4 rather than Equation 3. □

Again we note that, if the terminals are uniformly distributed over $[0,1]^d$, the integrals in Equations 9 and 10 equal 1.

For any dimension $d \geq 2$, we can calculate from Equation 9 the minimum t sufficient to achieve any desired performance ratio $B > 1$ as $n \to \infty$. Thus, if the terminals are uniformly distributed, we require $t \geq (\lambda/(B-1))^{d(d-1)}$, where λ is the constant factor in the big-Oh expression of Equation 9. For example, for this case with $d = 3$ and $B_3 = 1/0.813052 \approx 1.23$ under $\| \|_2$, Algorithm 1 attains a performance ratio smaller than B_3 when $t > (\lambda/(B_3 - 1))^6 \approx (4.35\lambda)^6$; this bound is of special interest in light of Smith's [28] disproof of the Gilbert-Pollak conjecture [16, 10] for $3 \leq d \leq 9$.

5 Conclusion

We have presented and probabilistically analyzed a deterministic partitioning approximation algorithm for the GSMT problem in R^d, for any dimension $d \geq 2$. Applying a theorem by Steele on subadditive Euclidean functionals, we proved the limit as $n \to \infty$ of the performance ratio of our Algorithm 1 is $1 + O(t^{-1/d(d-1)})$ almost surely, where t is a parameter the user can adjust to trade off time for performance. Our analysis holds for any norm on R^d, assuming the n terminals are any i.i.d. random variables taking on values in any bounded subset of R^d. The running time of Algorithm 1 is polynomial in n and, using the best available exact GSMT component algorithms, superpolynomial in t.

These results are significant because they yield a fast algorithm that has a guaranteed (with probability 1) worst-case performance ratio under more general assumptions than had been considered by Komlós and Shing [21]. The generalization to $d = 3$ under arbitrary norms on R^d is useful in multi-layer VLSI routing where wires are not restricted to be rectilinear.

Algorithm 1 can be easily extended to parallel and distributed versions. Partitioning is a natural parallel notion, and there are parallel algorithms for the other subroutines, such as computing medians and minimum spanning trees. Currently, we are implementing Algorithm 1 to measure its performance ratio experimentally.

This paper illustrates how the ideas of partitioning, approximation, and probabilistic analysis—augmented by the theory of subadditive Euclidean functionals—yield powerful tools for dealing with computationally difficult problems.

Acknowledgments

We thank Peter Matthews and Ravada Sivakumar for helpful editorial comments.

References

[1] A. V. Aho, M. R. Garey, and F. K. Hwang. Rectilinear Steiner trees: Efficient special-case algorithms. *Networks*, 7:37–58, 1977.

[2] A. Balakrishnan and N. R. Patel. Problem reduction methods and a tree generation algorithm for the Steiner network problem. *Networks*, 17:65–85, 1987.

[3] J. Beardwood, J. H. Halton, and J. M. Hammersley. The shortest path through many points. *Proceedings of the Cambridge Philosophical Society*, 55:299–327, 1959.

[4] M. W. Bern and R. L. Graham. The shortest-network problem. *Scientific American*, 260(1):84–89, January 1989.

[5] W. M. Boyce. An improved program for the full Steiner tree problem. *ACM Transactions on Mathematical Software*, 3(4):359–385, December 1977.

[6] E. J. Cockayne and D. E. Hewgill. Improved computation of plane Steiner minimum trees. *Algorithmica*, 7(2/3):219–229, 1992.

[7] E. J. Cockayne and D. G. Schiller. Computation of Steiner minimal trees. In Welsh and Woddall, editors, *Combinatorics*, pages 53–71. Inst. Math. Appl., 1972.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw Hill, Cambridge, Massachusetts, 1990.

[9] R. Courant and H. Robbins. *What is Mathematics?* Oxford University Press, New York, 1941.

[10] D.-Z. Du and F. K. Hwang. A proof of the Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica*, 7(2/3):121–135, 1992.

[11] H. Federer. *Geometric Measure Theory.* Springer-Verlag, New York, 1969.

[12] L. Few. The shortest path and the shortest road through n points. *Mathematika*, 2:141–144, 1955.

[13] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, July 1987.

[14] M. R. Garey, R. L. Graham, and D. S. Johnson. The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 32(4):835–859, June 1977.

[15] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, June 1977.

[16] E. N. Gilbert and H. O. Pollak. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29, 1968.

[17] M. Hanan. On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966.

[18] F. K. Hwang. On Steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, 30(1):104–114, January 1976.

[19] K. Kalpakis and A. T. Sherman. Probabilistic Analysis of an enhanced partitioning algorithm for the Steiner tree problem in R^d. Technical Report CS-TR-2936/UMIACS-TR-92-83, University of Maryland College Park, July 1992. Also available, in revised form, as Technical Report TR CS-92-10, Computer Science Department, University of Maryland Baltimore County.

[20] R. M. Karp. Probabilistic analysis of partitioning algorithms for the Travelling-Salesman Problem in the plane. *Mathematics of Operations Research*, 2(3):209–224, 1977.

[21] J. Komlós and M. T. Shing. Probabilistic partitioning algorithms for the rectilinear Steiner problem. *Networks*, 15:413–423, 1985.

[22] H. W. Kuhn. Steiner's problem revisited. In G. B. Dantzig and B. C. Eaves, editors, *Studies in Optimization. Studies in Mathematics, 10*, pages 52–70, Washington D.C., 1974. The Mathematical Association of America.

[23] Z. A. Melzak. On the problem of Steiner. *Canadian Mathematical Bulletin*, 4(2):143–148, May 1961.

[24] P. A. P. Moran. *An Introduction to Probability Theory.* Oxford University Press, London, 1968.

[25] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

[26] J. S. Provan. Convexity and the Steiner tree problem. *Networks*, 18:55–72, 1988.

[27] A. T. Sherman. *VLSI Placement and Routing: The PI Project.* Springer-Verlag, New York, 1989.

[28] W. D. Smith. How to find Steiner minimal trees in Euclidean *d*-space. *Algorithmica*, 7(2/3):137–177, 1992.

[29] J. M. Steele. Subadditive Euclidean functionals and nonlinear growth in geometric probability. *Annals of Probability*, 9(3):365–376, 1981.

[30] J. M. Steele. Probabilistic and worst case analyses of classical problems of combinatorial optimization in Euclidean space. *Mathematics of Operations Research*, 15(4):749–770, November 1990.

[31] D. Triesch and F. Hwang. An improved algorithm for Steiner trees. *SIAM Journal on Applied Mathematics*, 50(1):244–263, February 1990.

[32] P. Winter. An algorithm for the Steiner problem in the Euclidean plane. *Networks*, 15:323–345, 1985.

[33] P. Winter. Steiner problem in networks: A survey. *Networks*, 17:129–167, 1987.

On Solving Problems for p-Contractable Graphs *

Badrinath Ramamurthy and Mukkai S Krishnamoorthy
Department of Computer Science,
Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

In this paper we are concerned with some of the issues relating to the problem of embedding program graphs into a processor network architecture called the OPAM (Optical Parallel Architecture Model) introduced in [2]. We show here that the graph problem (which we call *bounded p-contractability*) related to embedding program graphs in this architecture is NP-complete. Our method can be used to show that for any fixed $p > 4$ (p is a parameter of the architecture,) the problem remains NP-complete. We also show by relating the class of bounded 2-contractable graphs to other well known classes of graphs[5], that Graph Isomorphism, Maximum Cut and Clique problems are in P for this class of graphs. We also provide an algorithm for finding the hamiltonian circuit in bounded 2-contractable graphs in linear time given the layout corresponding to a contraction.

1 Introduction

The OPAM architecture [2] consists of two levels of crossbar switches. Assume we have N processors. At the higher level is an NxN switch. At the lower level are k pxp cross bar switches each interconnecting a group of p processors and hence $N = kp$. The pxp crossbar switch operates at a much higher speed than the higher level crossbar. Each processor has a communication unit associated with it which provides a link to the higher level crossbar. Any processor within a group may use the link associated with any other processor in the group, but only active processors have their communication units active. A processor is active only if it has a process mapped to it. Thus, a link, with routing of messages within a given group, provides for p processors in one group to be connected to p processors in another group through the higher level cross bar.

In order for a process graph to be efficiently embeddable in this architecture, we want to know if there is a partitioning of the vertices of the graph so that there are no more than p vertices in each partition and the number of partitions a given partition is adjacent to is bounded by the number of vertices within that group. (Two partitions are adjacent if there is a vertex in one partition that is adjacent to a vertex in the other partition.) If a graph can be so partitioned, then we say that the graph is bounded p-contractable. More formally:

Definition: A graph G is said to be *bounded p-contractable* if there is a partitioning (called the *bounded p-contraction* of G) of its vertices into disjoint partitions of size at most p each such that for each partition the number of distinct partitions that its vertices are adjacent to is no larger than the number of vertices in that group.

The general problem is to see if a given graph G has a bounded p-contraction for a given integer p. We note that this problem is interestingly different from known NP-hard partitioning problems such as "minimum cut into bounded sets," or "graph partitioning," or "bin packing." Some other closely related problems are "achromatic number" and "domatic number." These problems are defined in [6].

---

*Supported in part by the NSF under grant CDA-8805910

2 Bounded p-Contractability is NP-complete

In this section we show that given a graph G and an integer p the problem of determining if G has a bounded p-contraction, which we shall refer to as Bounded p-Contractability, is NP-complete. In order to show this, we prove that the restriction where we set $p = 4$ is itself NP-complete by a reduction from 3-SAT.

Let us define our problems as below:

3-SAT: Given a set L of variables and a collection C of clauses over L such that each clause $c \in C$ has $|c| = 3$, is there a satisfying truth assignment for C ?

Bounded 4-Contractability: Given a graph G, does G have a bounded 4-contraction; that is, a partitioning of the vertex set $V(G)$ such that each partition has at most four vertices and is adjacent to no more partitions than it has vertices in it.

Outline of the construction

Here is the plan for the construction of our reduction from 3-SAT to Bounded 4-Contractability. We construct basic units to represent instances of a literal. The basic unit is such that it can be contracted in one of two ways: one to represent the literal being *True*, and the other for the literal being *False*. We link a set of basic units together into a "string" such that they all have to be contracted in the same way. Thus, all occurrences of the literal have the same truth value. Further, we interlink two strings such that the basic units in different strings are to be contracted in opposite ways. The two together represent to us instances of a variable and its negation. We have a vertex corresponding to each clause and connect it to a basic unit in each of the three strings corresponding to the literals in the clause. The aim of the interconnection is that the resulting graph must have a valid contraction if and only if the 3-SAT formula from which we started is satisfiable.

Construction

Given an instance (L, C) of 3-SAT, we now show our construction of a graph G corresponding to the instance. Let $|L| = n$ and $|C| = m$.

We first describe the basic unit H (see Figure 5) of our construction. H has four copies F, T, U and D of K_3, four copies $R_1, ..., R_4$ of the "forcing unit" of Figure 4, and vertices X, Y and Z interconnected as follows. Make F and T adjacent; i.e., connect each vertex of F with each vertex of T. Similarly make each of the following pairs adjacent: (T, U), (U, D), (X, F), (X, T), (Y, T), (Y, U), (Z, U), (Z, D), (X, Y) and (Y, Z). (Here we have loosely used X to also mean the set with the single vertex X. We shall do so as long as the meaning is clear.) We use the forcing unit R_1 by making it adjacent to F by which we mean connecting every vertex in F to every vertex in the set of vertices labeled "active end vertices" of R_1 as shown in Figure 4. one can show that this forces all vertices of F to be in the same partition in any valid contraction of the graph. Similarly we make D adjacent to R_2 and T adjacent to two forcing units R_3 and R_4.

For each literal l we interconnect a sequence of $m + 2$ basic units into another unit called a "string" for that literal. Let $H_0^l, ..., H_{m+1}^l$ be the $m + 2$ copies of H. We refer to the set of vertices F, T, X, etc., in H_i^l as F_i^l, T_i^l, X_i^l, etc. Connect U_i^l with U_{i+1}^l and Z_i^l with Z_{i+1}^l for $i \in 0..m$.

Interconnect strings corresponding to complementary literals l and $\bar{l}$ by connecting $U_0^{\bar{l}}$ and $Z_0^{\bar{l}}$ with F_0^l; and symmetrically U_0^l and Z_0^l with $F_0^{\bar{l}}$. This gives us an interconnected pair of strings for each of the n variables of L. Figure 6 shows a pair of strings corresponding to complementary literals.

For each clause c_i for $i \in 1..m$, if $c_i = (f, g, h)$ where f,g and h are literals, create a vertex called c_i and connect it to F_i^f, F_i^g and F_i^h.

Call the resulting graph G. An example is shown in Figure 7.

The Proof — Part I

We now prove that whenever C is satisfiable, G is bounded 4-contractable. Let S be an assignment of truth values to L. We will show how to construct a bounded 4-contraction for the graph G constructed above.

For each variable v such that $S(v) = True$, make the following partitions in the string corresponding to literal v: $X_i^v \cup T_i^v$, $Y_i^v \cup U_i^v$, and $Z_i^v \cup D_i^v$. F_i^v is left uncontracted for the moment. Do this for all instances of the unit in the string (i.e., for all $i \in 0..m+1$). Also in the string for $\overline{v}$, make the following partitions: $X_i^{\overline{v}} \cup F_i^{\overline{v}}$, $Y_i^{\overline{v}} \cup T_i^{\overline{v}}$, $Z_i^{\overline{v}} \cup U_i^{\overline{v}}$ and $D_i^{\overline{v}}$. Again, do this for all instances of the unit in the string (i.e., for all $i \in 0..m+1$).

Do the complementary action in the pair of strings for variables v such that $S(v) = False$.

Since for each clause at least one variable is true, it follows from the above construction that for each i, c_i is adjacent to F_i^l for some literal l such that F_i^l has not already been contracted. Make a partition with $\{c_i\} \cup F_i^l$.

For any F_i^l that have not already been contracted into a partition, set up a separate partition with F_i^l.

Contract each of the forcing units in the way shown in Figure 4.

That these steps yield a bounded 4-contraction can be verified easily.

The Proof — Part II

We now prove that each time the graph G has a bounded 4-contraction, the set of clauses C has a satisfying truth assignment.

The following lemmas form the body of our argument. For brevity sake, we leave out the proofs which can be found in [10].

Lemma 1 *In any valid contraction, for each literal l, all vertices of F_0^l, T_0^l, U_0^l and D_0^l occur together.*

Lemma 2 *In any valid contraction, for each literal l, one of the following holds:*

1. *X_0^l is partitioned with F_0^l, Y_0^l with T_0^l and Z_0^l with U_0^l (type* false*), or*

2. *X_0^l is partitioned with T_0^l, Y_0^l with U_0^l and Z_0^l with D_0^l (type* true*).*

(We name them so because units contracted in the first way will give an assignment of False *to the corresponding literal and similarly those contracted the second way will give us an assignment of* True *to the corresponding literal.)*

Lemma 3 *For a given literal l, for all $i \in 0..m$, the vertices of H_i^l are either all contracted type* true *or they are all contracted type* false.

Lemma 4 *In any contraction of G, for any variable l, both H_0^l and $H_0^{\overline{l}}$ cannot be of type* true.

Lemma 5 *In any bounded 4-contraction of G, the vertex c_i corresponding to any clause occurs in a partition with one of the F_i^g sets where g is a literal in c_i.*

Lemma 6 *If there is a bounded 4-contraction, then there is one with H_0^l and $H_0^{\overline{l}}$ contracted in different ways.*

Theorem 1 *If G is bounded 4-contractable, then the set of clauses C over the variables L has a satisfying truth assignment.*

Proof: Assume we have a contraction for G. By lemma 6 we can make a contraction in which the strings corresponding to complementary literals are contracted in opposite ways. Hence, we shall assume that the contraction we have has this property. We assign truth values to variables as follows: for each variable v if units in the string corresponding to literal v are contracted to the type *true*, then define $S(v) = True$, else define $S(v) = False$. Since all units in a string are contracted in the same way, this assignment is well defined. Also, it is consistent in the sense that for each variable l the negated and unnegated literal will have opposite truth values. Thus we have an assignment of truth values to variables of L.

We need to show now that this assignment satisfies S. Lemma 5 tells us that the vertex corresponding to clause $c_i = (f, g, h)$ is always with one of the F_i^l for $l \in \{f, g, h\}$, say F_i^g. This means that units in the string corresponding to literal g were contracted in the type *true* which means in the assignment we got, literal g was assigned *True*. This means that the corresponding clause c_i evaluates to true under this assignment. Since this holds for every clause c_i in C, the assignment satisfies C.

Thus our construction has a bounded 4-contraction each time the given instance of 3-SAT is satisfiable and conversely each time our graph is bounded 4-contractable, the 3-SAT instance is satisfiable. Hence, we have a valid reduction from 3-SAT to Bounded 4-Contractability. It is clear from the construction that the number of vertices and the number of edges in the constructed graph is $O(mn)$ so the reduction takes polynomial time. Thus Bounded 4-Contractability is NP-hard. Also, we can always verify if a given contraction is a valid bounded 4-contraction for the given graph in polynomial time. Thus the problem is in NP. Hence, we deduce that Bounded 4-Contractability is NP-complete.

From the above result, given a graph G and an integer p, the problem of deciding if G is bounded p-contractable is NP-complete.

3 Solving other problems for these graphs

Bounded k-contractable graphs appear to form a class of their own; i.e., they do not seem to be equivalent to well know subclasses of graphs. We now show how the class of bounded k-contractable graphs is related to some other classes of graphs and from that we draw conclusions about the difficulty of solving certain standard problems on these graphs. A discussion of the problems themselves can be found in [6].

- k-contractable graphs are bounded degree graphs, with a degree bound of $k^2 + k - 1$. This implies [5] that Clique and Graph Isomorphism are solvable in polynomial time for these graphs.
- Every 2-contractable graph has circular bandwidth ≤ 3. But for $k \geq 3$ k-contractable graphs do not have bounded circular bandwidth.
- For $k \geq 3$ a circular bandwidth k graph is k-contractable. All circular bandwidth 2 graphs are 3-contractable. All circular bandwidth 2 graphs on an even number of vertices are 2-contractable. All bandwidth 2 graphs are 2-contractable (to paths).
- 2-Contractable graphs have genus ≤ 1. This means that Maximum Cut is solvable in polynomial time for these graphs [5]. For k-contractable graphs, with $k > 2$, the genus is unbounded.

Next, we show that the problem of finding the hamiltonian cycle of a bounded 2-contractable graph is polynomial time solvable by simulating a wave front scanning across the 2-contraction of the graph. We observe that we can determine the chromatic number of these graphs using similar ideas. Since 2-contractable graphs have bounded circular bandwidth it may be possible to solve a class of other problems using this technique.

3.1 Hamiltonian Cycles using a moving wavefront

From [9] we see that a bounded 2-contraction for a graph gives us a either a cycle or a path. If the contraction is a cycle, all intermediate vertices of the contraction correspond to two vertices of the original graph. The same is true for all intermediate vertices for a path contraction. The end nodes of the path contraction may have either one or two vertices of the original graph.

We are interested in finding the hamiltonian cycle from a given 2-contraction. We say two edges are *independent* if they are not incident on a common vertex. If a path contraction of G is given to us, then it is easy to find the hamiltonian cycle using the property expressed in the following lemma.

Lemma 7 *Given a path contraction of G, G has a hamiltonian cycle* iff *(a) if an end group has only one vertex then that vertex is adjacent to both vertices in the group adjacent to it, (b) if an end node has two vertices, then there is an edge between the two vertices in that group, and (c) between every two groups containing two vertices each, there are two independent edges of G.*

Given a path contraction of G, lemma 7 gives us an $O(n)$ algorithm to determine if G has a hamiltonian cycle.

From now on we shall assume we are given the cycle contraction of a graph G. In Figure 1 we show three types of cuts we shall consider in the following treatment. Recall in these figures that the intended grouping is, by convention, $(a, c), (b, d), ...$ etc. In that figure applying cut 1 would remove any edges between a vertex on the shaded side and a vertex on the unshaded side.

We classify the possible hamiltonian cycles in cycle contractions into two types. Figure 2(a) illustrates type 1. These are characterised by the property that we would be able to *cut* the graph at some point by one of our cuts in such a way that none of the edges of a hamiltonian cycle would be in the cut. In the example, we could use a cut of type 2 between the left most and right most groups. Applying the cut would essentially remove all edges between the left most and right most groups. In contrast, Figure 2(b) illustrates a type 2 hamiltonian cycle. That is, the hamiltonian cycle has an edge crossing at every possible cut of type 1,2 or 3.

Definition: The path contractable graph resulting from the removal of edges crossing a certain cut at any point in G is called a *cut* graph.

Given a cycle contraction, we can check for a type 1 hamiltonian cycle by applying our cuts at all possible points in the graph, and checking for a hamiltonian cycle in each resultant cut graph. Since we need to consider at most $O(n)$ such cases, the algorithm takes time $O(n^2)$. In fact, we can derive an $O(n)$ algorithm by doing some preprocessing which takes only $O(n)$ time. We leave out the details due to space considerations.

If we were not able to detect hamiltonian cycles in the above two ways, then we look for a type 2 hamiltonian cycle in G. For the remainder of this section assume that the given cycle contraction of G has no type 1 hamiltonian cycle.

Lemma 8 *If G has a type 2 hamiltonian cycle then at every point in the contraction, one of the cuts at that point will have only one edge of the hamiltonian cycle crossing it.*

That the above lemma is true can be seen by enumerating all the eight possible ways the hamiltonian cycle can go between two adjacent groups of vertices. So all we need to do is check if for some cut at some arbitrary point there is an edge (x, y) crossing the cut such that the resultant cut graph has a hamiltonian path between the x and y. Clearly it is sufficient to check this for all edges crossing each of the three cuts at some fixed point. This means we need to check for hamiltonian paths only a constant number of times. A simple construction shows that each of the different hamiltonian paths that we need to find can be reduced to finding the hamiltonian path between vertex 1 and vertex n in a graph which is a subgraph (by edge removal) of some maximal 2-contractable graph M with even number of vertices as in Figure 3. We shall call such a hamiltonian path an *end to end* hamiltonian path (*ehp*).

We number the vertices of G by fixing the layout and then assigning the numbers $2i - 1$ and $2i$ to the vertices on the top and bottom "rows" inside the i^{th}group from the left, as we have done for M.

We move a wave front from left to right, computing partial hamiltonian paths at each step. Think of the *ehp* as being a directed path. Any edge along the path directed vertically or left to right crosses a cut of one of the types in Figure 1 so that all vertices to the left side of the cut have been already traversed by the path.

In Figure 1 let (b, d) be the i^{th} group from the left. Therefore the labels a, b, c and d are synonymous with $2i-3, 2i-1, 2i-2$ and $2i$ respectively. We shall refer to the cuts show in the figure as the i^{th} cuts of type 1,2 and 3.

The filled in vertices in the case of each cut represent *active* vertices, in the sense that if there is a hamiltonian path, which has covered all the vertices to the left of a cut(shaded region), then, it must come out of one of the active vertices and be incident on some vertex to the right of the cut. Given a cut of type $k (k \in 1..3)$ at a position i it is easy to compute the position of active vertices with respect to the cut. With respect to each cut we wish to maintain a log of whether there is a partial hamiltonian path ending at each active vertex. Let us say a partial hamiltonian path is *valid* with respect to the i^{th} cut of type k if it starts at vertex 1, traverses all vertices to the left of that cut and ends at a vertex that is active with respect to the cut.

Now we introduce the following notation. $A_k[i]$ is **true** if and only if there is a valid partial hamiltonian path with respect to the i^{th} cut of type k that ends at the vertex a, where a is defined for a given i as before. Since we look at only positions that are active with respect to a cut type, the only sequence of values we will compute are : A_1, B_1 and C_1 for cut type 1; B_2 and D_2 for cut type 2; and A_3, C_3 and D_3 for cut type 3. With this notation, we present the following algorithm to detect an *ehp* in the graph.

Algorithm EHP:

```
G: $n$ x $n$ Adjacency matrix of the graph with vertices numbered
    according to the layout. $G[i,j]$ is true $iff$ $(i,j) \in E(G)$
$A_1, B_1, C_1,$
$B_2, D_2,$
$A_3, C_3, D_3$; These are $n/2$ size arrays which will keep track of
    whether various cuts have valid partial hamiltonian paths ending
    at these positions.

    Initialise the arrays for the $1^{st}$ cuts of types 1,2 and 3;
    for $i \in 2..n/2$
    {
        $d := 2i; b := d-1; c := b-1; a := c-1;$   // convenient symbolic names
        $B_1[i] := B_2[i-1]$ && $G[a,b]$ || $D_2[i-1]$ && $G[c,b]$;
        $A_1[i] := D_3[i-1]$ && $G[c,b]$ && $G[b,a]$;
        $C_1[i] := B_1[i-1]$ && $G[a,b]$ && $G[b,c]$;

        $A_3[i] := D_3[i-1]$ && $G[c,d]$ && $G[a,d]$;
        $C_3[i] := B_1[i-1]$ && $G[a,d]$ && $G[c,d]$;
        $D_3[i] := B_2[i-1]$ && $G[a,d]$ || $D_2[i-1]$ && $G[c,d]$;

        $B_2[i] := A_3[i]$ && $G[a,b]$ || $D_3[i]$ && $G[b,d]$ || $C_3[i]$ && $G[c,b]$;
        $D_2[i] := B_1[i]$ && $G[d,b]$ || $A_1[i]$ && $G[a,d]$ || $C_1[i]$ && $G[c,d]$;
    }
end
```

To clarify things, the line "$A_1[i] := D_3[i-1]$ && $G[c,b]$ && $G[b,a]$;" says that for the i^{th} cut of type 1, there is a valid partial hamiltonian path ending at the vertex with position a relative to the cut (i.e. at vertex $2i-3$) if there is a valid hamiltonian path with respect to the

$(i-1)^{th}$ cut of type 3 ending at position d relative to that cut (i.e at $2(i-1)$) and if the edges (c,b) and (b,a) at point i are edges of G.

It is clear that at the end of the algorithm $D_2[n]$ is **true** if and only if in G there is a hamiltonian path from vertex 1 to vertex n. Since the algorithm moves the computation front through $n/2$ positions and at each position the computation takes no more than a constant time (this is the body of the above loop), the algorithm is $O(n)$. As mentioned earlier we need to do this for at most a constant number of times before deciding if the graph has a type 2 hamiltonian cycle. Thus we can find the hamiltonian cycle of a 2 contractable graph in $O(n)$ time given the 2-contraction. Building the 2-contraction takes time $O(|V|^3)$ based on the ideas presented in in [9].

4 Conclusion

We have shown that Bounded p-Contractability is NP-complete. We can use ideas similar to the one presented here to show that the problem remains NP-complete for any fixed $p > 2$. For $p = 2$, we have outlined a recognition algorithm in [9] that runs in $O(n^3)$ time where n is the number of vertices in the graph. We have related bounded p-contractable graphs to some well known classes of graphs. We also have provided an algorithm to detect hamiltonian circuits in bounded 2-contractable graphs.

References

[1] S.F. Assman, G.W. Peck, M.M. Syslo and J. Zak, *Bandwidth of caterpillars with hairs of length 1 and 2*, SIAM J. Alg. Discrete Meth. (1981), pp 387-391.

[2] A. Barak and E. Schenfeld, *Embedding Classical Communication Topologies in the OPAM Architecture*, Technical Report 90-12, Department of Computer Science, Hebrew University of Jerusalem, Israel.

[3] J. A. Bondy and U. S. R. Murty, *Graph Theory With Applications*, North-Holland, 1976.

[4] N. Deo, M. S. Krishnamoorthy and M. Langston, *Exact and Approximate Solutions for the Gate Matrix Layout Problem*, IEEE Trans. Computer Aided Design, 6, 1987, pp. 79-84.

[5] David S. Johnson, *The NP-Completeness Column: an Ongoing Guide*, Journal of Algorithms 6 (1985), pp 434—451.

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability*, W.H. Freeman and Company, NY, 1979.

[7] Joseph Y-T. Leung, O. Vornberger and J. D. Whitthoff, *On Some Variants of the Bandwidth Minimization Problem*, SIAM J. Computing, Vol. 13, No. 3, August 1984, pp 650—667.

[8] R. H. Möhring, *Graph Problems Related to Gate Matrix Layout and PLA Folding*, Computational Graph Theory, edited by G. Tinhofer, E. Mayr, H. Noltemeier and M. Syslo, Springer-Verlag, New York, 1990, pp. 17-51.

[9] B. Ramamurthy and M.S. Krishnamoorthy *Bounded 2-contraction of graphs*, in proceedings of the 1991 Annual Allerton Conference on Communication, Control and Computing, University of Illinois at Urbana-Champaigne, pp 498—507.

[10] Badrinath Ramamurthy and M.S. Krishanamoorthy, *Bounded p-Contractability is NP-complete*, Technical Report 92-17, Department of Computer Science, Rensselaer Polytechnic Institute, Troy-NY, June 1992.

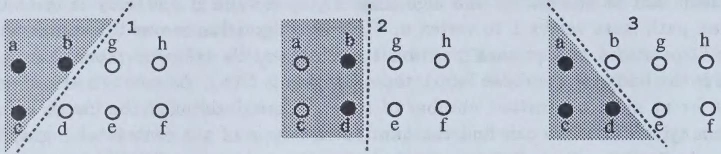

1
a
b
g
h
c
d
e
f
2
a
b
g
h
c
d
e
f
3
a
b
g
h
c
d
e
f

Figure 1: The three cuts

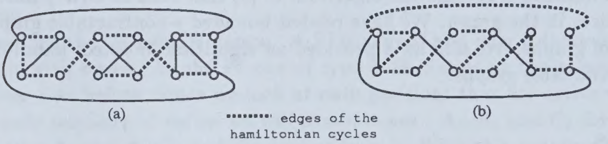

(a)
edges of the hamiltonian cycles
(b)

Figure 2: (*a*) Type 1 and (*b*) Type 2 Hamiltonian cycles

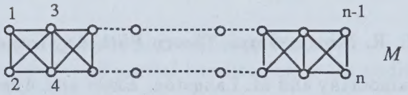


Figure 3: A maximal path bounded 2-contractable graph

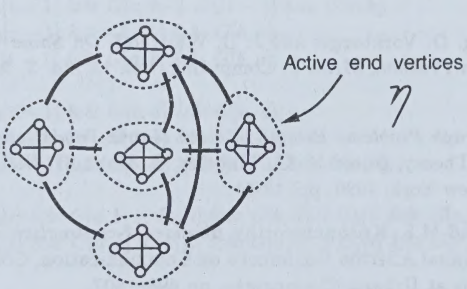

Active end vertices
η

Figure 4: The forcing unit R used with the basic unit in Figure 5. The heavy lines indicate multiple edges.

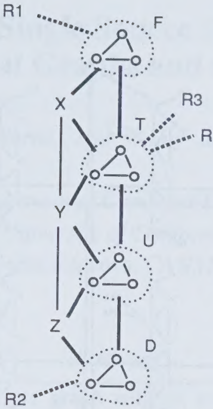

R1
F
X
R3
T
R4
Y
U
Z
D
R2

Figure 5: The basic unit H. The heavy lines indicate multiple edges. Broken heavy lines indicate multiple edges to the set η in a forcing unit (see Figure 4 for details of the forcing units.)

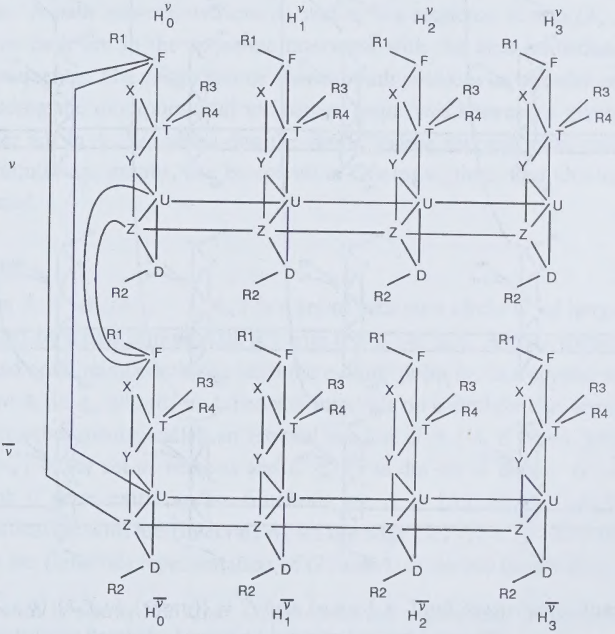
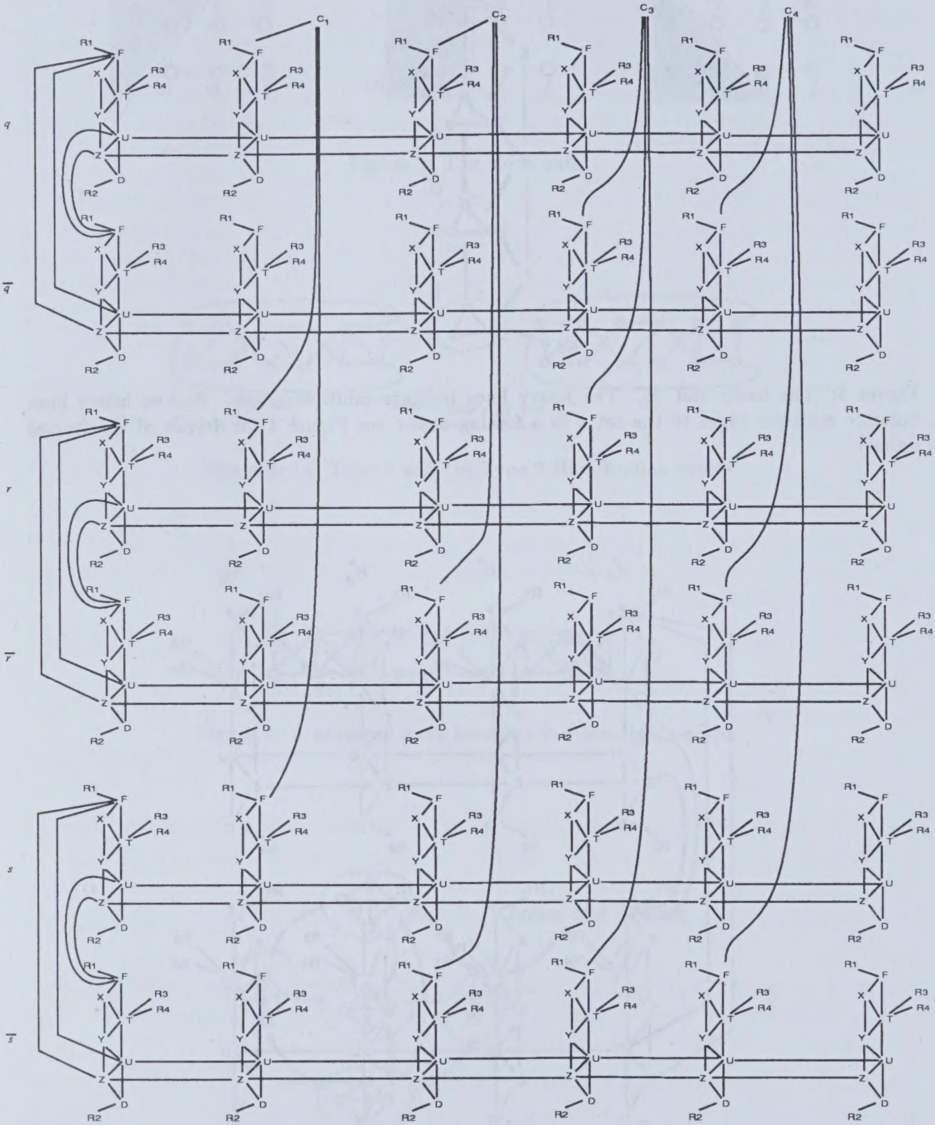

H^{v}_{0}
H^{v}_{1}
H^{v}_{2}
H^{v}_{3}
v
R1
F
X
R3
R4
T
Y
U
Z
D
R2
$\overline{v}$
$H^{\overline{v}}_{0}$
$H^{\overline{v}}_{1}$
$H^{\overline{v}}_{2}$
$H^{\overline{v}}_{3}$

Figure 6: Strings for complementary literals and their interconnection.



C1
C2
C3
C4
q
$\overline{q}$
r
$\overline{r}$
s
$\overline{s}$
R1
F
X
T
R3
R4
Y
U
Z
D
R2

Figure 7: Graph constructed for $L = \{q, r, s\}$ and $C = \{(q, r, s), (q, \overline{r}, \overline{s}), (\overline{q}, r, \overline{s}), (\overline{q}, \overline{r}, \overline{s})\}$.

Minimum Cover and Single Source Shortest Path Problems for Weighted Interval Graphs and Circular-arc Graphs*

Oscar H. Ibarra, Hui Wang† *and Qi Zheng*

Department of Computer Science
University of California
Santa Barbara, CA 93106

Abstract

Let $G = \{V, E\}$ be a circular-arc graph with an arc set representation A. A weight is associated with each arc in A. The minimum cover problem is to find a subset of arcs in A with minimum total weight whose union covers the whole circle. Here we present an $O(qn \log n)$ algorithm for the minimum cover problem on weighted circular-arc graphs, where q is the minimum number of arcs crossing any point on the circle. The previously best known algorithm runs in $O(qn^2)$. For weighted interval graphs, a subclass of weighted circular-arc graphs, our algorithm solves the minimum cover problem in the optimal $O(n \log n)$ time. A path between two arcs A_s and A_t is a sequence of arcs $(A_s, A'_1, A'_2, \cdots, A'_k, A_t)$, where each arc in the sequence intersects with the arcs immediately before and after it in the sequence. The single source shortest path problem on circular-arc graphs is the problem of finding the minimum total weight arc sequences between a given source arc A_s and every other arc in A. We show that the single source shortest paths problem, for both interval and circular-arc graphs, can be solved in $O(n \log n)$ time. Our $O(n \log n)$ time complexity is optimal.

1. Introduction

An arc set $A = \{A_1, A_2, \cdots, A_n\}$ is a set of arcs on a circle C of length L. Each arc A_i is represented by a pair of points (a_i, b_i) with $0 \le a_i, b_i \le L$. Arc A_i starts from the *starting* point a_i and continuing clockwise until the *ending* point b_i. In a special case where $a_i < b_i$ for every arc A_i in A, all arcs in A become intervals on a straight line segment $[0, L]$ and the arc set A is subsequently called an interval set. Let $G = \{V, E\}$ be a graph, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of vertices and $E \subseteq V^2$ is the set of edges. G is a circular-arc (interval) graph if there exists an arc (interval) set $A = \{A_1, A_2, \cdots, A_n\}$ such that arc (interval) A_i intersects with arc (interval) A_j iff the edge $(v_i, v_j) \in E$. The arc (interval) set A is called the arc (interval) representation of G, and A_i is the arc (interval) that corresponds

---

* Research supported in part by NSF Grant CCR89-18409 and a grant from the Institute for Scientific and Computation Research - Lawrence Livermore National Laboratory.

† Current address: Department of Computer Science, University of Alabama, Huntsville, AL 35899.

to vertex v_i, for $1 \leq i \leq n$. Without lose generality, we assume that all starting and ending points of arcs (intervals) in A are distinct.

Circular-arc and interval graphs are useful discrete mathematical structures for modeling practical problems in many areas such as biology, traffic control, VLSI design and job scheduling [4, 5]. Many sequential and parallel algorithms have been developed for solving combinatorial optimization problems on both graphs [2, 5, 6, 9, 11]. Two interesting problems concerning interval graphs are the minimum cover and shortest path problems [1, 3, 8, 10]. Let $A = \{A_1, A_2, \cdots, A_n\}$ be an arc representation of the circular-arc graph G where each A_i has an associated weight w_i. A set $S \subseteq A$ is called a *cover set* if the union of the arcs in S covers circle C. The *minimum cover (MC) problem* is to find a cover set S with minimum total weight. The *minimum cardinality cover (MCC) problem* is to find a cover set S with minimum cardinality. A path between arcs A_s and A_t is a sequence of arcs $P = (A'_1, A'_2, \cdots, A'_m)$ with each $A'_i \in A$ such that $A'_1 = A_s$ and $A'_m = A_t$, and A'_k intersects with A'_{k+1}, for $1 \leq k \leq m-1$. The length of P is the sum of the weights associated with arcs in the sequence. A path P between arcs A_s and A_t is a shortest path if P has the minimum length. The *shortest path (SP) problem* is defined as follows. Given $A_s, A_t \in A$, find a shortest path between A_s and A_t. The *single source shortest path (SSSP) problem* is the problem of finding the shortest paths between a given source arc A_s and every other arc in A. Correspondingly, we can define the MC, MCC, SP and SSSP problems for interval graphs with interval representations.

Lee and Lee [8] gave an O($n\log n$) time algorithm for the MCC problem. Later, Atallah and Chen [1] provided a parallel algorithm on an EREW PRAM for the MCC problem which runs in O($\log n$) time on O(n) processors. The previously best known sequential algorithm for the MC problem on circular-arc graphs, given by Bertossi [3], runs in O(qn^2) time where q is the minimum number of arcs crossing any point of the circle C. Bertossi raised the question whether a faster algorithm exists for the MC problem on circular-arc graphs. In this paper, we present an O($qn\log n$) algorithm for solving this problem.

As result of the minimum cover problem, we show that both SP and SSSP problems can be optimally solved in O($n\log n$) time for weighted circular-arc graphs and interval graphs. Olariu et. al. [10] gave an O($n\log n$) algorithm for the SP problem on unweighted interval graphs. With some modifications, the algorithms in [1] and [8] can be used to solve the SP problem on unweighted circular-arc graphs. It was shown in [10] that SP and SSSP problems have time complexities lower bound $\Omega(n\log n)$ [10]. To the best of our knowledge, the algorithms for SP and SSSP problems presented in this paper are the first such algorithms that achieve the optimal O($n\log n$) time complexity for both weighted interval and circular-arc graphs.

In the next section, we give some definitions and observations concerning interval graphs. Section 3 presents the algorithms for the MC, SP and SSSP problems on interval graphs. Section 4 contains the algorithms for these problems on circular-arc graphs.

2. Preliminaries

Let $A = \{A_1, A_2, \cdots, A_n\}$ be the interval representation of an interval graph G. Each element A_i, represented by a pair of starting and ending points (a_i, b_j), is an interval on the line segment $[0, L]$. We assume that all starting/ending points of n intervals are distinct.

With a simple sorting, we can relabel the intervals in A according to the increasing order of the starting points, i.e., $i < j$ iff $a_i < a_j$. An interval set A is called *sorted* if the intervals in A are labeled according to the increasing order of their starting points.

Two intervals A_i and A_j intersect with each other iff they share at least one common point on the line segment $[0, L]$, i.e., $A_i \cap A_j \neq \emptyset$. If $A_i \cap A_j = A_j$ and $i \neq j$, then A_j is *contained* in A_i and such A_j is called a *contained interval*.

We define a function *successor* as in [1, 8], $SUC: A \rightarrow A$, as follows: $SUC(A_i) = A_j$ iff $j = \max\{ k \mid a_i \leq a_k < b_i \}$. In other words, $SUC(A_i) = A_j$ iff a_j is the last starting point encountered by a left to right sweep from a_i to b_i. The function values $SUC(A_i)$, for $1 \leq i \leq n$, can be computed in $O(n \log n)$ time [1, 8]. For the completeness of presentation, we given a sketch here. First, sort the starting and ending points of n arcs in A in increasing order $(x_1, x_2, \cdots, x_{2n})$. Assign a weight to each of these x's as $w(x_i) = a_j$ if $x_i = a_j$ and $w(x_i) = -\infty$ otherwise. $SUC(A_i) = A_j$ if $a_j = \max\{x_1, x_2, \cdots, x_k\}$ for $x_k = b_j$. It is possible that $SUC(A_i) = A_i$.

Note that in a sorted interval set A, interval A_k intersects with interval A_i with $k > i$ iff $SUC(A_i) = A_j$ and $i < k \leq j$. This special property will be used in our algorithms.

3. The Minimum Cover and Shortest Path Problems on Interval Graphs

In this section we first present an algorithm **INTERVAL** for the problem of finding the minimum cover for weighted interval sets. Then we show how to use **INTERVAL** to solve the shortest path and single source shortest paths problems on weighted interval graph.

Let $A = \{A_1, A_2, \cdots, A_n\}$ be the interval set and $A_i = (a_i, b_i)$ with $a_i < b_i$ for $1 \leq i \leq n$. A weight $w_i \geq 0$ is associated with each interval A_i. Without loss of generality, we assume that $a_1 = 0$. Let $max\_b = \max\{b_1, b_2, \cdots, b_n\}$ and A_t be the interval with $b_t = max\_b$. A_t and max_b can be found in linear time. A subset S of A is a minimum cover of A iff S has the minimum total weight and the union of intervals in S covers the line segment $[0, max\_b]$. Since all starting and ending points are distinct, it is obvious that A_1 and A_t must be in the minimum cover S.

Lemma 3.1. A set S is a minimum cover iff the intervals in S form a shortest path between A_1 and A_t.

Proof. Straightforward and omitted. □

To find the minimum cover of an interval set A, we only need to find a shortest path between interval A_1 and interval A_t.

Let $P(i)$ denote the shortest path from A_1 to interval A_i such that $P(i) = (A'_1, A'_2, \cdots, A'_m)$ with $A'_k \in A$ for $1 \leq k \leq m$, $A'_1 = A_1$ and $A'_m = A_i$. The length of path $P(i)$ is $D(P(i)) = \sum_{k=1}^{m} w'_k$. For a given point p on the real line $[0, max\_b]$, we say a path $P(i)$ can reach point p iff $b_i \geq p$. Let the set $R(p)$ be the set of shortest paths which can reach point p. Define $MIN(p) = \{ P(i) \mid D(P(i)) = \min \{D(P(j)) \mid P(j) \in R(p)\}\}$. In other words, path $P(i)$ is in $MIN(p)$ iff $P(i)$ is the shortest path from A_1 to interval A_i and $P(i)$ is a path with minimum length that reaches point p. Since $w_i \geq 0$ for $1 \leq i \leq n$, if an interval A_j intersects with A_1 then the path $P(1) = (A_1)$ is in $MIN(a_j)$.

Lemma 3.2. Let A be an interval set. A path $P = (A'_1, A'_2, \cdots, A'_m)$, with $A'_1 = A_1$ and $A'_m = A_i$, is a shortest path from A_1 to A_i iff subpath $(A'_1, A'_2, \cdots, A'_j)$ is in $MIN(a'_{j+1})$ for $1 \leq j < m$.

Proof. Since P is a path, A'_1 (i.e., A_1) must intersect with A'_2, then $P(1) = (A'_1) = (A_1)$ is in $MIN(a'_2)$.

Suppose that P is a shortest path from A_1 to A_i and there exists a subpath $S = (A'_1, A'_2, \cdots, A'_j)$ which is not in $MIN(a'_{j+1})$. S is in $R(a'_{j+1})$ because A'_j intersects with A'_{j+1}. Since S is not in $MIN(a'_{j+1})$, there must exist a path Q such that Q is in $MIN(a'_{j+1})$ and $D(Q) < D(S)$. By replacing S with Q, the resulting path $P' = Q \cdot (A'_{j+1}, \cdots, A'_m)$ will be a shorter path than P. This contradicts the fact that P is the shortest path from A_1 to A_i.

Now suppose that subpath $(A'_1, A'_2, \cdots, A'_j)$ is in $MIN(a'_{j+1})$ for $1 \leq j < m$ and P is not the shortest path from A_1 to A_m. Let $Q = (B_1, B_2, \cdots, B_s)$ be the shortest path from A_1 to A_i. We can see that $B_1 = A'_1 = A_1$ and $B_s = A'_m = A_i$. Let k be the smallest integer such that $B_{s-k} \neq A'_{m-k}$. In other words, $B_s = A'_m$, $B_{s-1} = A'_{m-1}, \cdots, B_{s-k+1} = A'_{m-k+1}$ and $B_{s-k} \neq A'_{m-k}$. If such k does not exist, then $Q = P$ and P is also a shortest path. Suppose such k exists, then both subpaths $(A'_1, A'_2, \cdots, A'_{m-k})$ and $(B_1, B_2, \cdots, B_{s-k})$ are in $R(a'_{m-k+1})$. Since $(A'_1, A'_2, \cdots, A'_{m-k})$ in $MIN(a'_{m-k+1})$, we have $D((A'_1, A'_2, \cdots, A'_{m-k})) \leq D((B_1, B_2, \cdots, B_{s-k}))$. Hence, the length of path P cannot be longer than the length of path Q. Therefore, P must be a shortest path from A_1 to A_i, which contradicts our earlier assumption. □

We describe below the algorithm **INTERVAL** that solves the minimum cover problem for an input interval set A by finding the shortest path from A_1 to A_t.

Algorithm **INTERVAL**:

Input: An interval representation A of an interval graph.
Output: The minimum cover set of A.

```
Begin
(1)  Sort $A$ and relabeling each interval according to the increasing order of starting points.
(2)  Find $A_t$ with $b_t = max\_b$.
     If $t = 1$ then the minimum cover $S$ is $\{A_1\}$ and stop.
(3)  Computes $SUC(A_i)$, for $1 \leq i \leq n$.
(4)  Let $m$ be the index of $SUC(A_1)$.
     If $m < 2$ then report no cover and Stop.
```

```
(5) $PEND = \varnothing$
    For $k = 2$ to $m$ do
        (* Arc $A_k$ directly intersects with $A_1$ *)
        $D_k = w_1 + w_k$
        $P(k) = (A_1) \cdot (A_k)$
        $PEND = PEND \cup (k, D_k, -b_k)$
    od
(6) Sort $PEND$ on the key $(D_i, -b_i)$ in nondecreasing order and keep the sorted heap.
(7) while $m \le t$ and $PEND \ne \varnothing$ do
        Find $(j, D_j, -b_j)$
        (* Such that $(j, D_j, -b_j)$ is the minimum element in $PEND$. *)
        Let $r$ be the index of $SUC(A_j)$.
        Remove $(j, D_j, -b_j)$ from $PEND$.
        If $r > m$ then
            For $u = m+1$ to $r$ do
                $D_u = w_u + D_j$.
                Insert $(u, D_u, -b_u)$ into $PEND$.
                $P(u) = P(j) \cdot (A_u)$.
            od
            $m = r$
        fi
    end of while
(8) If $m \ge t$ then
        The minimum cover $S = P(t)$.
        The weight of $S = D_t$.
    else
        Report no cover.
    fi
end.
```

Lemma 3.3. Algorithm **INTERVAL** finds the minimum cover for an interval set A.

Proof. We only need to prove that the cover set S found by algorithm **INTERVAL** is a shortest path from A_1 to A_t (i.e., $P(t)$ is a shortest path from A_1 to A_t). Stage (1), (2) and (3) are preprocessing stages. Their functions and correctness are given in Section 2. We use a min-heap called *PEND* (path end) to save information about all paths computed by the algorithm. Each element in *PEND* is a triplet $(k, D_k, -b_k)$, where k is the index of computed path $P(k)$ from A_1 to A_k, D_k is the length of $P(k)$, and b_k is the largest reachable point by $P(k)$. Point b_k is recorded as $-b_k$ for the convenience of heap sorting used later. If a triplet $(k, D_k, -b_k)$ is *PEND*, then the corresponding path $P(k)$ is called *active path.*

Proposition 3.1. If path $P(k)$ is an *active path,* then algorithm **INTERVAL** has found the path $P(i)$ which is a shortest path from A_1 to A_i for all $2 \le i \le k$.

Proof. We prove the proposition by induction on the construction of *PEND*. In stage (4),

INTERVAL finds interval A_m which is *SUC*(A_1). If $m = 1$, then A_1 does not intersect with any other intervals and there is no path between A_1 and A_t. Hence, no minimum cover S exists. If $m > 1$, then for $1 < k \leq m$, interval A_k directly intersects with A_1 because Stage (1) converts A to a sorted interval set. It is obvious that $P(k) = (A_1, A_k)$ is the shortest path between A_1 and A_k. The proposition is true for the initial construction of *PEND*. From now on, the membership of *PEND* can only be changed in stage (7). Let us assume that the proposition is true after the k-th iteration of the **while**-loop in stage (7). Let m be the largest index in *PEND* and $(j, D_j, -b_j)$ be the triplet that has the smallest D_j such that b_j is the largest. In other words, the path $P(j)$ reaches the point b_j with the minimum length D_j among the lengths of all active paths. In the $(k+1)$-th iteration, **INTERVAL** first finds *SUC*$(A_j) = A_r$. If $r \leq m$, then $P(j)$ cannot be used to find any new shortest path because all of the shortest path $P(i)$ for $i \leq m$ have been found. Triplet $(j, D_j, -b_j)$ is removed from *PEND*. In the case $r > m$, new path $P(u) = P(j) \cdot (A_u)$ is computed for $m < u \leq r$. To show that $P(u)$ is the shortest path from A_1 to A_u, it suffices to show that $P(j)$ is in *MIN*(a_u) because $P(j)$ is the shortest path from interval A_1 to interval A_j.

Suppose that a path $P(i)$ is in *MIN*(a_u) and $P(j)$ is not in *MIN*(a_u) [i.e., $D(P(j)) = D_j > D(P(i))$]. We show that such $P(i)$ does not exist. First of all, $P(i)$ cannot be an *active path* because D_j is minimum among all the lengths of *active paths*. We consider three different cases where $i \leq m, m < i < u$, or $i \geq u$.

Suppose $i \geq u$. If $P(i)$ is in *MIN*(a_u), a path $P(i')$, with $i' < u$, should also be in *MIN*(a_u). Since $u \leq i$ and A is a sorted interval set, we have $a_u \leq a_i$. Because $P(i)$ is a shortest path from A_1 to A_i, there must exist an interval $A_{i'}$ in $P(i)$ and $A_{i'}$ intersects with A_u for $1 \leq i' < u$, Subpath from A_1 to $A_{i'}$ in $P(i)$ must be $P(i')$ which is the shortest path from A_1 to $A_{i'}$. It is not difficulty to see that both $P(i)$ and $P(i')$ can reach point a_u. If $P(i)$ is in *MIN*(a_u), then $P(i')$ is also in *MIN*(a_u) because $D(P(i')) \leq D(P(i))$.

Now consider the case $i \leq m$. Since $P(i)$ cannot be an *active path*, the triplet $(i, D_i, -b_i)$ must have been deleted from *PEND* before the $(k+1)$-th iteration. $P(i) \cdot (A_u)$ cannot be a path because A_i does not intersect with A_u. If A_i intersects with A_u, then path $P(u)$ must be found before the $(k+1)$-th iteration.

Now we only need show that $P(i)$ cannot exist for $m < i < u$ and $P(j)$ is in *MIN*(a_u) for $m < u \leq r$. If $u = m+1$, it is easy to see that such $P(i)$ does not exist and $P(j)$ is in *MIN*(a_{m+1}). Suppose that $P(j)$ is in *MIN*(a_u) for all u with $m+1 \leq u < r$. By this assumption, path $P(i) = P(j) \cdot (A_i)$ should be a shortest path from A_1 to A_i for $m < i < r$. If $P(i)$ is in *MIN*(a_r) then $P(j)$ is also in *MIN*(a_r), because $D(P(j)) \leq D(P(i)) = D(P(j)) + w_i$ for $m+1 \leq i < r$.

We have shown that $P(j)$ is in *MIN*(a_u) which implies that $P(u) = P(j) \cdot (A_u)$ is a shortest path from A_1 to A_u for $m < u \leq r$. Now r is the largest index in *PEND*. For $1 \leq i \leq r$, shortest path $P(i)$ has been found by algorithm **INTERVAL**. Triplet $(j, D_j, -b_j)$ is deleted from *PEND* because after the $(k+1)$-th iteration path $P(j)$ has been extended to all of the intervals which A_j intersects with. Path $P(j)$ cannot directly reach any new intervals after the $(k+1)$-th iteration. □

(Proof of Lemma 3.3. continued) In stage (8), **INTERVAL** checks whether $P(t)$ is an *active path* (i.e., whether the largest index m in *PEND* is larger than t). If $m \geq t$, then by Proposition 5.1, the shortest path from A_1 to A_t is found. If $m < t$, the interval graph is not a connected graph and no path exists between A_1 and A_t. Algorithm **INTERVAL** correctly finds the shortest path $P(t)$ between A_1 and A_t. Therefore, it correctly finds the minimum cover for interval set A. □

Lemma 3.4. Algorithm **INTERVAL** runs in $O(n \log n)$ time for an input interval set A of size n.

Proof. Clearly, the time complexity of algorithm **INTERVAL** is dominated by stages (1), (6), and (7). All other stages in the algorithm **INTERVAL** need no more than $O(n)$ time. Stage (1) and (6) can be carried out by sorting which needs $O(n \log n)$ time [7, 12]. Stage (7) consists of a **while**-loop with a **for**-loop inside of it. Since r is bounded by the size of the interval set A and each interval is handled by the **for**-loop at most once, the number of iterations of the **for**-loop is bounded by n. The total time needed by the **for**-loop is $O(n \log n)$ since the insertion operation on a heap needs $O(\log n)$ time [7, 12]. One can easily see that the rest of the computations of the **while**-loop are bounded by $O(n \log n)$ time. Hence, the total time needed by algorithm **INTERVAL** is $O(n \log n)$. □

Theorem 3.1. The minimum cover problem for interval set can be solved in optimal $O(n \log n)$ time.

Proof. The theorem follows from Lemmas 3.3, 3.4 and the time complexity lower bound proof in [8]. □

With careful analysis of algorithm **INTERVAL**, we can see that **INTERVAL** not only finds the shortest path from A_1 to A_t but also finds all the shortest paths from A_1 to A_k with $1 < k \leq t$. If we let A_t be the interval with largest starting point (i.e., A_n), **INTERVAL** can finds all of the shortest paths from A_1 to every other interval in A. Actually with some modifications on interval set A, **INTERVAL** can solve both the shortest path and single source shortest path problems for weighted interval graphs.

Theorem 3.2. The shortest path and single source shortest path problems for weighted interval graphs with interval representations can be solved in optimal $O(n \log n)$ time.

Proof. Given interval graph G with interval representation set A, let A_i and A_j be two intervals in A. If A_i directly intersects with A_j, the shortest path between A_i and A_j is (A_i, A_j). Otherwise, assume that $a_i < a_j$, let $a_i = 0$ (forcing $A_i = A_1$) and apply algorithm **INTERVAL** with $A_t = A_j$ on A. Obtained path $P(t)$ is the shortest path between A_i and A_j. Since the modification on A only requires constant time, the shortest path problem on weighted interval graphs can be solved in $O(n \log n)$ time.

Given an interval A_i in A, to find the shortest paths from A_i to every other interval in A, we use algorithm **INTERVAL** twice. First, let $a_i = 0$ and use **INTERVAL** on A with $A_t = A_n$. After this, we found the shortest paths between A_i and A_k with $i < k$. Then, let $b_i = max\_b$ and modify **INTERVAL** such that it finds shortest paths from right to left instead of from left to right (the usual direction). Apply the new **INTERVAL** on modified A to find the shortest paths from A_i to A_k with $k < i$. Clearly the above steps are enough for finding all the shortest paths starting from A_i. The time used is no more than $O(n \log n)$. Hence, the single source shortest paths problem on weighted interval graphs can be solved in O($n \log n$) time. The optimality follows from the time complexity lower bound established by [10]. □

4. The Minimum Cover and Shortest Path Problem on Circular-arc Graphs

Let $A = \{A_1, A_2, \cdots, A_n\}$ be an arc representation of a circular-arc graph G. Each A_i is an arc on a circle C of length L. Let $S(A_i)$ denote a minmum cover set which contains arc A_i. $S(A_i)$ may not be the minimum cover set S for A. If the arcs in $S(A_i)$ are ordered by increasing starting points, then two necessary conditions for $S(A_i)$ to be the minimum cover set S of A are that (i) no arc in $S(A_i)$ is contained in union of any other arcs in $S(A_i)$ and (ii) every two (cyclically) consecutive arcs in $S(A_i)$ are intersected while nonconsecutive arcs do not intersect at all [3]. A minimum cover set $S(A_i)$ satisfying the above two conditions is called a *useful* $S(A_i)$. Next, we show how to find useful $S(A_i)$ for given A_i.

We solve the problem of finding the *useful* $S(A_i)$ by reducing it to a shortest path problem on an interval graph $G(A_i)$ with interval representation I. Let W be the set of arcs in $A - \{A_i\}$ which contains a_i, the starting point of arc A_i. We can assume that $a_i = 0$. If $a_i \neq 0$, we can rotate the circle C and force $a_i = 0$. The rotation needs only linear time. If A_j is in W then $a_j < L$ and $0 < b_j$ with L and 0 representing the origin of circle C. If A_j is in $A - W$, we have $a_j < b_j$. Now we show how to create the interval set I. First, for each arc $A_j \in A - W$ and $j \neq i$, let $I(A_j)$ be an interval (a_j, b_j) with weight w_j in the interval set I. For each arc A_j in W, we convert A_j to an interval $I(A_j) = (a_j, L)$ with weight w_j in I. Finally, let intervals $I(A_i) = (a_i, b_i)$ with weight w_i and $I'(A_i) = (L, L)$ with weight 0 be the two intervals in I for arc A_i.

Lemma 4.1. A minimum cover set $S(A_i)$ is *useful* iff $S(A_i)$ corresponds a shortest path from $I(A_i)$ to $I'(A_i)$ in the interval set I.

Proof. Sort the arcs in $S(A_i)$ according their starting points to a sequence $(A_i, A'_1, A'_2, \cdots, A'_{m-1}, A'_m)$. If the corresponding sequence $(I(A_i), I(A'_1), \cdots, I(A'_m), I'(A_i))$ is a shortest path from $I(A_i)$ to $I'(A_i)$, then it is obvious that $S(A_i)$ is *useful*. Now suppose that $S(A_i)$ is *useful*. Then A'_k is not in W for $1 \leq k \leq m-1$. Assume that there is a $A'_k \in W$ and $k \neq m$. In the sorted sequence, we have $a'_k < a'_j < L$ for $k < j \leq m$. If A'_j is not in W then $b'_j < L$. Therefore, A'_k contains A'_j and $S(A_i)$ is not *useful*. If A'_j is in W then both A'_k and A'_j intersect with A_i and $S(A_i)$ is not *useful* either.

We see that if any arc is in $W \cap S(A_i)$ and $S(A_i)$ is useful, then it must be the last arc in the sorted sequence. For any arc A_j in W, the part of arc $(0, b_j)$ does not contribute to the computation of $S(A_i)$ at all. Therefore, the correctness of mapping the arc set A to the interval set I follows immediately. The sorted sequence corresponds to a shortest path $(I(A_i), I(A'_1), \cdots, I(A'_m), I'(A_i))$ in I. □

Theorem 4.1. For any given arc A_i in an arc set A, the useful minimum cover set $S(A_i)$ can be found in O($n \log n$) time.

Proof. Lemma 4.1. states that $S(A_i)$ can be found by first mapping the arc set A to an interval set I and then finding the shortest path between $I(A_i)$ and $I'(A_i)$. The mapping uses no more than O(n) time. In section 3, we showed that the shortest path problem on a weighted interval graph can be solved in O($n \log n$) time. □

The *usefulness* is the necessary condition for $S(A_i)$ to be the minimum cover set S of A, but it is not sufficient. Let q be the minimum number of arcs crossing any point on the circle C and Q be the set of those arcs. For $A_i \in Q$, there must exist at least one useful $S(A_i)$ which is the minimum cover S of A.

Theorem 4.2 The minimum cover set S for an arc set A can be found in O($qn \log n$) time.

Proof. The set Q can be found in O(n) time [3]. For each $A_i \in Q$, finds the useful $S(A_i)$. The minimum cover set S of A is the $S(A_i)$ which has the minimum weight. □

To solve the shortest path and single source shortest path problems on circular-arc graphs, we reduce them to corresponding problem on interval graphs. Here we only show that the single source shortest path problem on circular-arc graphs can be solved in O($n \log n$) time. Since the shortest path problem is contained in the single source shortest path problem, it can be solved in O($n \log n$) time too.

Theorem 4.3. The single source shortest path problem on circular-arc graphs with arc representations can be solved optimally in O($n \log n$) time.

Proof. Let G be a circular-arc graph with arc representation A. For any given arc A_i (assume that $a_i = 0$) in A, let W be the set of arcs in $A - \{A_i\}$ which contains a_i. The shortest path $P(A_j)$ between A_i and A_j is (A_i, A_j) for any $A_j \in W$ because they are directly intersected. For any A_j which is not in W, the shortest path $P(A_j)$ is the shorter path between the shortest clockwise path from A_i to A_j (denoted as $SCP(A_j)$), and the shortest counter-clockwise path from A_i to A_j (denoted as $SCCP(A_j)$).

To compute $SCP(A_j)$ for all $A_j \in A-W$, we map the arc set A to an interval set I_{SCP} by breaking circle C at point a_i (i.e., point 0). For every arc $A_k \in W$, let $(0, b_k)$ with weight w_k be the interval representing A_k in I_{SCP}. For every arc $A_j \in A-W$, let (a_j, b_j) with weight w_j be the interval for A_j in I_{SCP}. What the mapping did is cutting off the $(a_k, 0)$ part of every arc $A_k \in W$.

We prove that discarding the $(a_k, 0)$ part of every arc $A_k \in W$ does not effect our computation for the shortest paths. Let $A_j \in A-W$. Suppose that the $SCP(A_j)$ must use the $(a_k, 0)$ part of an arc $A_k \in W$. Then $SCP(A_j)$ must be in the form of $(A_i, \cdots, A_k, \cdots, A_j)$. Since $SCP(A_j)$ is the shortest clockwise path from A_i to A_j and $(a_k, 0)$ is in $SCP(A_j)$, a_j must be greater than a_k (i.e, $a_k < a_j < L$). Therefore, A_k intersects with A_j. The sequence (A_i, A_k, A_j) is a counter-clockwise path from A_i to A_j and it is no longer than $SCP(A_j)$. Hence, $SCP(A_j)$ can not be the shortest path $P(A_j)$. The mapping dose not effect the computation for the shortest paths $P(A_j)$.

Using algorithm **INTERVAL** (given in section 3) on I_{SCP}, we can find all $SCP(A_j)$ for $A_j \in A-W$. The time used by **INTERVAL** is O($n \log n$). The mapping only uses linear time. The total time used for finding the shortest clockwise paths is no more than O($n \log n$).

Since the way of finding $SCCP(A_j)$ is symmetric to the way of finding $SCP(A_j)$ for all $A_j \in A-W$, we leave it to reader. □

Corollary 4.1. The shortest paths problem on circular-arc graphs with arc representations can be solved optimally in $O(n \log n)$ time.

Proof. Follows from Theorem 4.3.

5. Conclusion

We have presented the algorithms for the shortest path, single source shortest path, and minimum cover problems for both weighted interval and circular-arc graphs. Our algorithm for the minimum cover problem on weighted circular-arc graphs require $O(qn \log n)$ time which is faster than the $O(qn^2)$ algorithm proposed by [3]. When input is unsorted, the $O(n \log n)$ running time is optimal for the shortest path problem on both interval and circular-arc graphs and the minimum cover problem on interval graphs. It is interesting to know whether faster algorithm can be achieved for these problems if the input is presorted.

References

1. Atallah, M. and D. Z. Chen, An Optimal Algorithm for the Minimum Circle-Cover Problem, *Information Processing Letters*, 32, 1989, pp. 159-165.
2. Bertossi, A. A. and M. A. Bonuccelli, Some Parallel Algorithms on Interval Graphs, *Discrete Applied Mathematics*, 16, 1987, pp. 101-111.
3. Bertossi, A. A. Parallel Circle-Cover Algorithms, *Information Processing Letters*, 27, 1988, pp. 133-139.
4. Booth, K. S. and G. S. Lukeher, Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using PQ-Tree Algorithms, *J. of Computer and System Science*, 13, 1976, pp. 335-379.
5. Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
6. Gupta, U. I., D. T. Lee and J. Y.-T. Leung, Efficient Algorithms for Interval Graphs and Circular-Arc Graphs, *Networks*, Vol. 12, 1982, pp. 459-467.
7. Knuth, D. E., *The Art of Computer Programming*, Vol.3, Addison-Wesley, 1973.
8. Lee, C. C. and D. T. Lee, On a Circle-Cover Minimization Problem, *Information Processing Letters*, 18, 1984, pp. 109-115.
9. Lu, T.-L., P.-H. Ho and G. J. Chang, The Domatic Number Problem in Interval Graphs, *SIAM J. Dis. Math.*, Vol. 3. No. 4, 1990, pp. 531-536.
10. Olariu, S., J. L. Schwing and J. Zhang, Optimal Parallel Algorithms for Problems Modeled by a Family of Intervals, *Proceeding of the 28th Allerton Conference*, 1990. pp. 282-291.
11. Ramkumar, G. D. S. and C. P. Rangan, Parallel Algorithms on Interval Graphs, *Proceedings of the 1990 International Conference on Parallel Processing*, Vol. 3, 1990, pp. 72-74.
12. Williams, J. W. J., Heapsort, *CACM*, 7, 1964, pp. 347-348.

PALINDROMIC TREES

Bettina De Iaco [1], **Fabrizio Luccio** [2] **and Linda Pagli** [2]
[1] Istituto di Informatica, University of Parma
Cornocchio, 43100 Parma, Italy
[2] Dipartimento di Informatica, University of Pisa
Corso Italia 40, 56100 Pisa, Italy

Abstract

Palindromic labelled trees (paltrees) are defined, in the binary and general case. Under a string representation of trees, paltrees become special palindromes (t-palindromes) of the string. Several properties of t-palindromes are proved, allowing to derive a simple algorithm to find all the palindromic subtrees of a given tree in linear time and space.

1. Position of the problem

The recognition of palindromes in strings (i.e., substrings that read the same backwards and forwards) has been treated extensively. In particular, Manacher [6] and Galil [3] have provided linear time algorithms to determine the initial palindromes in a string. The problem has then been solved in real-time by Galil [4]. Crochemore and Rytter have shown how to determine all the palindromes of even length in O(logn) parallel time with n processors [1]. Knuth, Morris and Pratt [5] recognize the language palstar (concatenation of palindromes) in linear time. In this paper we extend the concept of palindromes to ordered labelled trees. Some theoretical results lead to a linear algorithm to determine all the palindromic subtrees of a given tree. In the present and the following sections all trees are binary, while the case of general ordered trees is discussed in section 4.

Formally, let α be a binary tree, Σ be an alphabet and $\sigma(N) \in \Sigma$ be the *label* associated to node N of α. The following definitions are taken from [2].

Definition 1. The *reverse* α^r of a binary tree α, is a binary tree such that:

1. $\alpha=\alpha^r=\phi$,

or

2. $$\alpha = \begin{array}{c} N \\ / \quad \backslash \\ \beta \qquad \gamma \end{array}, \quad \alpha^r = \begin{array}{c} N' \\ / \quad \backslash \\ \gamma^r \qquad \beta^r \end{array}, \quad \sigma(N)=\sigma(N').$$

Definition 2. A binary *palindromic tree* (shortly *paltree*) α is such that:

$$\alpha = \begin{array}{c} N \\ / \quad \backslash \\ \beta \qquad \beta^r \end{array}.$$

Note that, if α is a paltree, $\alpha^r=\alpha$.

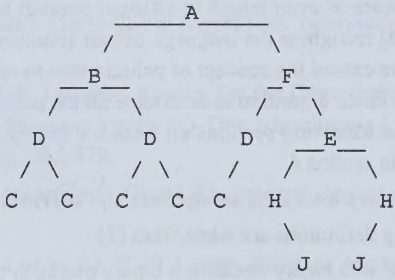
For example, in the tree of fig.1 the two subtrees rooted at the sons of the node labelled E are one the reverse of the other. The subtree rooted at E is a paltree.

We represent trees as strings, and study the properties of the substring associated to paltrees (section 2). Based on these properties, we devise an algorithm to determine all the paltrees of a given binary tree of n nodes (section 3), which requires O(n) time and space. The extension of our study to arbitrary ordered trees is treated in section 4.

2. Properties of paltrees

A labelled and binary tree α of n nodes is represented with a string S, obtained by an inorder traversal of α. Let us number the tree nodes from 1 to n, according to the order in which they are visited. The p-th element of S is associated to node p, $1 \le p \le n$, and consists of the pair $(\sigma(p),\tau(p))$, where $\sigma(p)$ is the label of node p and $\tau(p)$ is its level (the root has level 1). See fig.1.

For $1 \le i \le j \le n$, S(i,j) denotes the substring of S from position i to position j. Note that a subtree β of α is associated with a substring S(h,k) of S. If β is rooted in r, $h \le r \le k$, we denote $S^*(r)=S(h,k)$, $\lambda(r)=h$ (*left terminator*), $\rho(r)=k$ (*right terminator*). If a node b is contained in the subtree rooted in r, the substring $S^*(b)$ is contained in $S^*(r)$. Note that the functions σ, τ, λ, ρ can be computed during the tree traversal with overall linear complexity. The following properties 1 and 2 derive from [2].



position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
label σ	C	D	C	B	C	D	C	A	C	D	F	H	J	E	J	H
level τ	4	3	4	2	4	3	4	1	4	3	2	4	5	3	5	4
left term. λ	1	1	3	1	5	5	7	1	9	9	9	12	13	12	15	15
right term. ρ	1	3	3	7	5	7	7	16	9	10	16	13	13	16	15	16

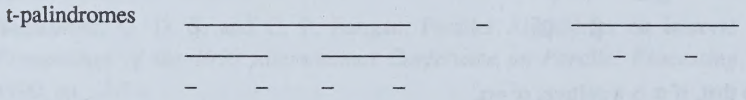


Fig.1 A binary tree α and its string representation

Property 1. If the subtree rooted in r is a paltree, $S^*(r)$ is a palindrome.
Proof. Immediate.

For example, nodes 14 and 15 in fig.1 are the roots of two paltrees, associated to the palindromes $S(12,16)$ and $S(15,15)$, respectively. Note that no palindrome in S may have even length because any two adjacent values of τ must be different. A palindrome in S does not necessarily correspond to a paltree: see the palindrome $S(6,10) = (D,3)\ (C,4)\ (A,1)\ (C,4)\ (D,3)$. A palindrome in S corresponding to a paltree is called a *t-palindrome*. The segments at the bottom of figure 1 show the spans of all the t-palindromes. We have:

Property 2. A palindrome $T = S(i,j)$ is a t-palindrome if $\lambda(m) = i$ and $\rho(m) = j$ with $m = (i+j)/2$ (that is, T codes a subtree rooted in m).
Proof. (i) Let W_1, W_2 be two substrings coding two subtrees γ_1, γ_2, and let W_1 be the reverse of W_2. By an easy induction on the heigth of γ_1 (or γ_2) it can be proved that $\gamma_2 = \gamma_1^r$.
(ii) By hypothesis T codes the subtree β rooted at m. Hence the substrings $T_1 = S(i,m-1)$ and $T_2 = S(m+1,j)$ recursively code the subtrees β_1, β_2 of β, rooted at the sons of m. Since T is a palindrome, T_1 is the reverse of T_2, hence $\beta_2 = \beta_1^r$ by point (i). Therefore β is a paltree and T is a t-palindrome. Q.E.D.

Property 1 shows that the problem of finding all the paltrees in a given tree is solved by finding all the t-palindromes in S, that is, finding all the palindromes that fulfill the conditions of property 2. This could be achieved by determining, for each position k of S, the palindrome of maximal length (*max-palindrome*) centered in k, and controlling if this max-palindrome encompasses a t-palindrome also centered in k.

The max-palindromes of S could be found by a simple variation of Manacher's algorithm [6]. (Indeed, the algorithm of [6] determines only the max-palindromes of even length, spanning to the left end of S, while we search for max-palindromes of odd length and arbitrary span). However we introduce a simpler algorithm based on some properties of t-palindromes. We also benefit from the fact that t-palindromes code subtrees, hence they cannot partially overlap (while palindromes can). For this purpose we need the following results.

Property 3. If a substring $S(h,k)$ is the coding of a subtree β, we have $\tau(h-1)\neq\tau(k+1)$, $\tau(h-1)<\tau(i)$ and $\tau(k+1)<\tau(i)$, $h\leq i\leq k$. (If $h-1=0$ or $k+1=n+1$, the corresponding conditions are implicitly satisfied).
Proof. Immediate from the properties of inorder traversal. Letting r be the root of β, note that the elements of S in positions $h-1$ and $k+1$ (if any) code two ancestors of r at different levels ($\tau(h-1)\neq\tau(k+1)$, $\tau(h-1)<\tau(r)$, $\tau(k+1)<\tau(r)$), and that $\tau(r)<\tau(i)$ for $h\leq i\leq k$ and $i \neq r$. Q.E.D.

The next property states that a substring $S^*(m)$ coding a subtree cannot be properly contained in the max-palindrome centered in m.

Property 4. If $S(i,j)$ is a max-palindrome centered in $m = (i+j)/2$, we have $i\geq\lambda(m)$ and $j\leq\rho(m)$.
Proof. Let $m-\lambda(m)\leq\rho(m)-m$, and let $i<\lambda(m)$ by contradiction (the case $m-\lambda(m)>\rho(m)-m$, $j>\rho(m)$ is similar). The elements of S in positions $\lambda(m)-1$, $2m-(\lambda(m)-1)$ are internal to $S(i,j)$, therefore they must be equal because $S(i,j)$ is a palindrome. However, the element in $\lambda(m)-1$ is left adjacent to the substring $S^*(m)$, and the element in $2m-(\lambda(m)-1)$ is right adjacent to $S^*(m)$, for $m-\lambda(m) = \rho(m)-m$, or internal to $S^*(m)$, for $m-\lambda(m) < \rho(m)-m$. In both cases the two elements must have different values of τ by property 3. Q.E.D.

Note that if $i=\lambda(m)$ and $j=\rho(m)$ in property 4, then $S(i,j)$ is a t-palindrome by property 2. Let us now examine how max-palindromes and t-palindromes may overlap.

Property 5. Let $S(i,j)$ and $S(h,k)$ be max-palindromes, respectively centered in $m = (i+j)/2$ and $r = (h+k)/2$, and let $i \leq r < m$. Then there exists another max-palindrome $S(u,v)$ centered in $w = 2m-r$ such that:

1) $v = 2m-h$, for $h>i$;
2) $v \geq 2m-h = j$, for $h=i$;
3) $v = j$, for $h<i$.

Proof. For a given position x, denote by $\underline{x} = 2m-x$ the symmetrical position with respect to m.
1) $h>i$. The substring $S(u,v) = S(\underline{k},\underline{h})$ is contained in $S(i,j)$, therefore is the reverse of the palindrome $S(h,k)$. Then $S(u,v)$ is a palindrome. Moreover the elements in positions $h-1$, $k+1$ are different because $S(h,k)$ is a max-palindrome, therefore the elements in positions $\underline{k}-1$, $\underline{h}+1$ are different. That is $S(u,v)$ is a max-palindrome.
2) $h=i$. As for case 1), $S(\underline{k},\underline{h}) = S(\underline{k},j)$ is a palindrome, but not necessarily a max–palindrome since the elements in positions $j+1, j+2\ldots$ may be equal to the ones in positions $\underline{k}-1, \underline{k}-2\ldots$.
3) $h<i$. Let $z=2r-i$. The string $S(i,z) \subset S(h,k)$ has a palindromic reverse $S(u,v) = S(\underline{z},j)$. To prove that this is also a max-palindrome, consider the four elements a,b,c,d in position $i-1$, $z+1$, $\underline{z}-1$ and $j+1$, respectively. We have $a=b$ and $b=c$ (symmetrical pairs inside the palindromes $S(h,k)$ and $S(i,j)$, respectively). Furthermore $a \neq d$ because they are adjacent to the max-palindrome $S(i,j)$. Therefore $d \neq c$, that is, $S(\underline{z},j)$ is a max–palindrome. Q.E.D.

Note that, in property 5, the center w of $S(u,v)$ is symmetrical to the center r of $S(h,k)$ with respect to m. Let us now specialize this property for t-palindromes. For a max-palindrome $S(x,y)$ centered in $z = (x+y)/2$, let the *radius* be defined as $R(z) = z-x$ (max-palindromes of length one have radius zero). We have:

Property 6. Let $S(i,j)$ be a t-palindrome centered in $m = (i+j)/2$, and $S(h,k)$ be a max-palindrome centered in $r = (h+k)/2$, with $i \leq r < m$. Then there exists another max-palindrome $S(u,v)$ centered in $w = 2m-r$ such that $R(w) = R(r)$. Furthermore, if $S(h,k)$ is a t-palindrome, then $S(u,v)$ is a t-palindrome.

Proof. Immediate from properties 4 and 5, and the observation that t-palindromes code subtrees.

In the next section we show how the properties given above can be used to build a simple and efficient algorithm to determine all the paltrees contained in a binary tree.

3. The basic algorithm

The determination of all the paltrees of a given binary tree α amounts to finding all the t-palindromes in the string S representing α (properties 1,2). This is obtained with a crucial use of the other properties of section 2. In fact, our algorithms detects the t–palindromes as a restricted family of max-palindromes. With the notation of property 5, the radius $R(w)$ of the max-palindrome $S(u,v)$ can be computed in constant time, once $R(m)$, $R(r)$ are known, thus avoiding repeated comparisons of elements of S. $R(w)$ is computed by the function RADIUS reported below. In case 2) of property 5, however, the computation may be completed by further character comparisons outside the span of $S(i,j)$ (this situation is indicated by the value *true* of the boolean variable extend). The text of RADIUS may be better understood noting that $r = 2m-w$, $h = r-R(r) = 2m-w-R(2m-w)$, $i = m-R(m)$.

```
function RADIUS(m,w):
 if 2m-w-R(2m-w)≥m-R(m)
   then RADIUS:=R(2m-w)                          {cases 1), 2) of property 5}
   else RADIUS:=2m-w-(m-R(m));                   {case 3) of property 5}
 if 2m-w-R(2m-w)=m-R(m) then extend:=true else extend:=false
end RADIUS.
```

The following procedure PALTREE computes all the t-palindromes of S. Two terminators S(0)=$ and S(n+1)=@ are added to S, with $≠@, and both elements different from all the other elements in S.

```
procedure PALTREE:
e:=h:=1;
for k:=1 to n do
 if (k≤e-1 and L(h)≠0)                                  {property 6}
   then
     begin L(k):=L(2h-k); R(k):=R(2h-k)
     end
   else
     begin
       if k≤e-1
         then
           begin
             R(k):=RADIUS(h,k);                         {property 5}
             if extend=true then                        {case 2) of property 5}
               while S(e)=S(2k-e) do
                               {increase R(k) by direct character comparisons}
                 R(k):=R(k)+1; e:=e+1
               endwhile
           end
         else
           begin
             R(k):=0; e:=k+1;
             while S(e)=S(2k-e) do
               R(k):=R(k)+1; e:=e+1
             endwhile
           end;
       if (λ(k)+ρ(k))/2=k and k-λ(k)=R(k))              {properties 2 and 5}
         then L(k):=2R(k)+1
         else L(k):=0;
       h:=k
     end
endfor
end PALTREE.
```

The output of PALTREE is reported in a vector L(1:n), such that L(i) contains the length of the t-palindrome centered in i. (If the t-palindrome is empty we set L(i):=0). Two pointers k,e scan S from left to right. k points to the center of the max-palindrome (possibly a t-palindrome) K under construction, and e points to the rightmost element examined thus far. Another pointer h<k points to the center of the max-palindrome P previously computed, which spans most to the right. If k falls within P, and P is a t-palindrome, the value of L(k) is computed in constant time by letting L(k):=L(2h-k) (property 6). If k falls within P and P is not a t-palindrome, a preliminary value of R(k) is computed in constant time by the function RADIUS. Then, if case 2) of property 5 occurs, the final value of R(k) is computed by direct comparisons of the elements in positions e and 2k-e, for increasing e. If k does not fall within P, the value of L(k) is also computed by direct comparisons of the elements on the two sides of k. Since k and e never move backwards, we attain linear time. Formally we pose:

Theorem. All the t-palindromes of S can be computed in O(n) time and space.

Proof. Examine the procedure PALTREE and the function RADIUS, noting that the sequences σ, τ, λ, ρ can be preliminarly computed in linear time and space. Q.E.D.

4. Paltrees with nodes of arbitrary degree

In the previous section we have shown how all the palindromic subtrees of a given labelled binary tree can be determined in linear time and space. Let us now extend our definitions and results to the case of labelled ordered trees whose nodes have arbitrary degree (called simply "trees" in the following).

Definition 3. The *reverse* α^r of a non empty tree α, is a tree such that:

1. if α consists of a single node, then: $\alpha=N$, $\alpha^r=N'$, $\sigma(N)=\sigma(N')$;

2. if the root of α has $k\geq 1$ sons, then:

$$\alpha = \begin{array}{c} N \\ / \quad\quad \backslash \\ \beta_1 \cdots \beta_k \end{array}, \quad \alpha^r = \begin{array}{c} N' \\ / \quad\quad \backslash \\ \beta_k{}^r \cdots \beta_1{}^r \end{array}, \quad \sigma(N)=\sigma(N').$$

Definition 4. A *palindromic tree* α (shortly *paltree* if no ambiguity arises with the binary case) is such that either α consists of a single node, or:

$$\alpha = \begin{array}{c} N \\ / \quad\quad \backslash \\ \beta_1 \cdots \beta_k \end{array}, \text{ with } \beta_i = \beta^r_{k+1-i}, \quad 1 \leq i \leq \lfloor (k+1)/2 \rfloor .$$

Definition 3 includes definition 1 as a special case, if we complete the binary tree with empty external nodes in place of the empty sons. Then, definition 4 includes definition 2 as a special case. It can be immediately shown that, if α is a paltree, $\alpha^r=\alpha$. Note that in definition 4 we have $\beta_{(k+1)/2} = \beta^r_{(k+1)/2}$ for k odd. For example, the tree of fig.2 is a paltree. The two subtrees with roots labelled B are one the reverse of the other. The central subtree with root labelled C is a paltree.

As for binary trees, we represent a tree α of n nodes with a new combinatorial structure S consisting of a string with emerging spikes which are in turn strings. Let $\sigma(R)$, $\tau(R)$ be the label and the level of node R. We have:

Definition 5. A tree (or subtree) α is represented with a *spiked-string* S_α such that:

1. if α consists of a single node R, then $S_\alpha=(\sigma(R),\tau(R))$;

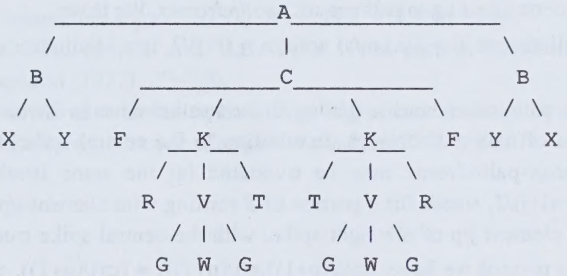
2. if $\alpha = \begin{array}{c} R \\ / \qquad \backslash \\ \beta_1 \;\cdots\; \beta_k \end{array}$, then:

$$S_\alpha = S_{\beta_1} \dots S_{\beta_{k/2}}\ (\sigma(R),\tau(R))\ S_{\beta_{(k/2+1)}} \dots S_{\beta_k}, \quad \text{for k even,}$$

$$S_\alpha = S_{\beta_1} \dots S_{\beta_{(k-1)/2}}\ \underset{(\sigma(R),\tau(R))}{S_{\beta_{(k+1)/2}}}\ S_{\beta_{(k+3)/2}} \dots S_{\beta_k}, \quad \text{for k odd,}$$

where, if a subtree β has root Q, the notation: $\underset{(\sigma(R),\tau(R))}{S_\beta}$ denotes that $(\sigma(R),\tau(R))$ is placed below $(\sigma(Q),\tau(Q))$ in an emerging spike of S_α.

An example of spiked-string S is shown in fig.2. The structure S can be built in linear time with a tree traversal. A string position p is associated to each element (σ,τ) of the base chain of S. If a spike T emerges from p, position p is associated to T and each element of T is associated with the pair p/p', where p' is the position of the element in T. (All the elements of



position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
σ, τ	X3	B2	Y3	F3	R4	G5	W5	G5	T4	C2	T4	G5	W5	G5	R4	F3	Y3	B2	X3
							V4			A1			V4						
							K3						K3						
λ	1	1	3	4	5	6	7	8	9	4	11	12	13	14	15	16	17	17	19
ρ	1	3	3	4	5	6	7	8	9	16	11	12	13	14	15	16	17	19	19
							6			1			12						
							8			19			14						
							5						11						
							9						15						

Fig.2 A tree of arbitrary degree and the corresponding spiked–string S

the chain of S have p'=1, that will be omitted. Obviously, S can be seen as a sparse array with two indices). For example, in fig.2, the position of the spike (W,5) (V,4) (K,3) is 7; the position of element (F,3) is 4/1 or simply 4; the position of element (A,1) is 10/2.

While S codes a whole tree α, a subtree β of α is coded with a substring from i to j, where the spike of the root may be taken only in part. In fig.2, the whole portion of S between positions 5 and 9 codes the subtree rooted in K, while the portion between 6 and 8, with the spike in 7 limited to first two elements (W,5) (V,4), codes the subtree rooted in V. In general, S(i,j), $1 \le i \le j$, denotes the portion of S from position i to j with complete spikes; S(i,j,h/k), $1 \le i \le h \le j$, $k \ge 1$, denotes the portion of S from i to j with the spike in h limited to the first k elements. Therefore, S(5,9) and S(6,8,7/2) code the two subtrees mentioned above.

As in the binary case, the tree nodes will be identified with their positions p/p' in S. If a subtree β is associated with S(h,k,r/s), we denote such a portion of S by $S^*(r/s)$, and let $\lambda(r/s)=h$, $\rho(r/s)=k$ (see fig.2). If a node u/v is contained in the subtree rooted in r/s, $S^*(u/v)$ is a substructure of $S^*(r/s)$. Note that u/v and r/s may belong to the same spike, that is u=r, v<s. The functions λ,ρ can be computed in linear time during the tree traversal. We now define a palindrome on S as if S was a string, with spikes treated as single elements (some spikes may be truncated as indicated). Properties 1 and 2 are then extended to the following properties 7 and 8 (given without proofs).

Property 7. If the subtree rooted in r is a paltree, $S^*(r/s)$ is a palindrome.

In fig.2, the subtree rooted at 10/1 (node C) has $S^*(10/1)$ = F3 R4 G5 (W5 V4 K3) G5 T4 C2 T4 G5 (W5 V4 K3) G5 R4 F3. Note that the central spike is truncated at C2.

Palindromes in S corresponding to paltrees are *t-palindromes*. We have:

Property 8. A palindrome T = S(i,j,m/s) with m = (i+j)/2, is a t-palindrome if $\lambda(m) = i$ and $\rho(m) = j$.

Finding all the paltrees amounts to finding all the t-palindromes in S,which in turn requires the determination of max-palindromes. In addition to the central spike, the spikes at the extremes of a max-palindrome may be truncated (at the same level). The notation S(i/p,j/p,m/q), m=(i+j)/2, stands for a portion of S starting with element i/p of the left spike and ending with element j/p of the right spike, with the central spike truncated at m/q. If S(i/p,j/p,m/q) is a maxpal we have: $(\sigma(i/(p+1)),\tau(i/(p+1))) \neq (\sigma(j/(p+1)), \tau(j/(p+1)))$, if the spikes at i, j are longer than p; $(\sigma((i-1)/1),\tau((i-1)/1)) \neq (\sigma((j+1)/1), \tau((j+1)/1))$ otherwise.

Properties 3, 4, 5 and 6 can be extended to the present case. In fact, the following property 9, derived from property 5, is crucial to attain linear time in the determination of t-palindromes.

Property 9. Let S(i/p,j/p,m/q) and S(h/t,k/t,r/s) be max-palindromes, with $i \le r < m$. Then there exists another max-palindrome S(u/x,v/x,w/y) with w = 2m-r such that:

1) v = 2m-h and x = t, for h>i, or h=i and t<p;
2) v = 2m-h and $x \ge t$, or $v \ge 2m-h$, for h=i and t=p;
3) v = j and x = p, for h<i, or h=i and t>p;

(the values of q, s, y are immaterial).

The following property 10, derived from property 6, allows to speed up the search for t-palindromes.

Property 10. If S(i,j,m/q) and S(h,k,r/s) are t-palindromes, with $i \le r < m$, then there exists another t-palindrome S(u,v,w/s) with w = 2m-r and the same radius of S(h,k,r/s).

The procedure PALTREE of section 3 can be extended to determine all the t-palindromes in a spiked-string, in a natural although non trivial way. Instead of the simple pointers h, k, e, three pairs h/h', k/k', e,e' are used to point to positions in S. The structure is scanned by increasing k/k' to k/(k'+1), if the latter position exists, or to (k+1)/1 otherwise. The pair e/e' is increased similarly. Since each pair assumes at most n different values, we attain linear time. Formally we pose:

Corollary. All the t-palindromes of a spiked-string S can be computed in O(n) time and space.

Acknowledgment

This work has been supported in part by M.U.R.S.T. of Italy.

References

[1] M. Crochemore and W. Rytter. Usefulness of Karp-Miller-Rosenberg algorithm in parallel computations on strings and arrays. *Th. Comp. Sci.*88 (1991) 59-82.

[2] B. De Iaco and F. Luccio. Finding all the palindromes in a binary tree in linear time and space. Dipartimento di Informatica, Pisa. Technical Report TR 28/91 (1991).

[3] Z. Galil. Real time algorithms for string matching and palindrome recognition. *Proc. 8-th ACM STOC* (1976) 161-173.

[4] Z. Galil. Two fast simulations which imply some fast string matching and palindrome-recognition algorithms. *Inf. Proc. Lett.* 4 (1976) 85-87.

[5] D.E. Knuth, J.H. Morris and V.R. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*6 (1977) 323-350.

[6] G. Manacher. A new linear-time on-line algorithm finding the smallest initial palindrome of a string. *J. Assoc. Comput. Mach.*22 (1975) 346-351.

Fast Algorithms for Inferring Evolutionary Trees

RICHA AGARWALA, DAVID FERNÁNDEZ-BACA, AND GIORA SLUTZKI
Department of Computer Science, Iowa State University, Ames, IA 50011

Abstract

We present algorithms for the perfect phylogeny problem restricted to binary characters. The first algorithm is faster than a previous algorithm by Gusfield when the input matrix for the problem is sparse. Next, we present two online algorithms. For the first of these, the characters are given as input one at a time, while, for the second, the species are given as input one at a time. These two online algorithms can easily be combined into an algorithm that can process any sequence of additions and deletions of species and characters.

1 Introduction

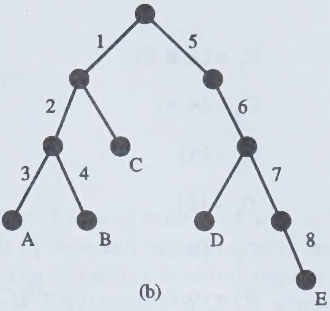
A fundamental problem in molecular biology is that of inferring the evolutionary history of a set of species, each of which is specified by the set of *traits* or *characters* that it exhibits [3, 4]. Each character can occur in a species in one of a fixed number of *states.* Information about evolutionary history can be conveniently represented by an evolutionary or *phylogenetic* tree, often referred to simply as a *phylogeny.* More precisely, a phylogeny for a set of species $\mathcal{S}$ is a rooted tree in which the leaves represent the species in $\mathcal{S}$, and the internal nodes represent hypothetical ancestral species. The species are usually described by an $n \times m$ matrix M, where $n = |\mathcal{S}|$ is the number of species, m is the number of characters, and M_{ij} is the state of the j^{th} character for the i^{th} species. A *perfect phylogeny* P, if it exists, is a phylogeny that assigns an m-vector of character states to each hypothesized ancestral species, and has the property that, for each state of each character, the set of nodes in P having that state is connected in P. Recently, this problem was shown to be NP-complete by Bodlaender et. al. [1].

When we restrict the characters to have only two states, M becomes a 0-1 matrix and an alternative but equivalent representation of P is a rooted tree where each species in $\mathcal{S}$ is attached to exactly one leaf of P, each character is associated with exactly one edge of P, and for any leaf w of P, the characters associated with the edges along the unique path from root to w exactly specify the characters of the species at leaf w. See Figure 1 for an example. We shall refer to this special case of perfect phylogeny problem as the *binary phylogeny (BP)* problem. For the biological assumptions and interpretation of this problem, see [5].

Gusfield [5] gave a time-optimal algorithm for solving BP which takes time linear in the size of the input matrix M. As he points out, the argument for the $\Omega(nm)$ lower bound considers a very unnatural matrix in which most of the species have most of the characters and M has $nm-2$ ones and only 2 zeroes. In general, however, M is not dense. For such instances, the following representation of the input may be more natural: For each character $i \in \{1, \cdots, m\}$, we are given a set C_i of all the species that exhibit that

	1	2	3	4	5	6	7	8
A	1	1	1	0	0	0	0	0
B	1	1	0	1	0	0	0	0
C	1	0	0	0	0	0	0	0
D	0	0	0	0	1	1	0	0
E	0	0	0	0	1	1	1	1

(a)



(b)

Figure 1: (a) A matrix M and (b) its phylogenetic tree P.

character; i.e., C_i is the set of species which have a 1 in column i of M. For convenience, we shall assume the existence of a universal set of species C_0 containing all n species. From now on, we shall refer to C_i as a character. We assume that all characters are nonempty. Our first result is a $O(C)$ algorithm for BP where $C = \sum_{i=1}^{m} |C_i|$. This is a significant improvement over the $O(nm)$ algorithm given in [5] for the case where M is sparse. Next, we give two online algorithms. The first one receives characters as input one at a time. We suggest two implementations of this algorithm and show that depending on the implementation, the running time of the algorithm for adding a character C_t is $O(|C_t| \log n)$ or $O(n)$. When invoked repeatedly for adding m characters, this yields a total running time of $O(C \log n)$ or $O(nm)$. The second online algorithm receives species as input one at a time. Similar implementations as those of the previous algorithm result in running time of $O(|I(S_t)| \log m)$ or $O(m)$ where S_t is the species being added and $|I(S_t)|$ is the number of ones in the row for S_t in M. This leads to a total running time of $O(C \log m)$ or $O(nm)$ for adding n species. Since, as pointed out in Sections 3 and 4, online deletion of characters and species from a given phylogeny can also be done quickly, we have a complete set of primitives for maintaining a phylogeny under any stream of additions and deletions of characters and species.

2 An algorithm for BP

Let $\mathcal{C} = \{C_0, \cdots, C_m\}$ be a set of characters. Gusfield's algorithm for BP and ours are based on the following fundamental result.

Lemma 1 *[2] $\mathcal{C}$ has a perfect phylogenetic tree P iff for every pair of characters C_i, C_j, either $C_i \cap C_j = \emptyset$ or one contains the other.*

A set of characters $\mathcal{C}$ satifying the conditions of this lemma shall be said to be *compatible.*

Our approach is based on constructing a *character tree (CT)* T for $\mathcal{C}$. T is a rooted tree with the following properties:

(C1) $V(T) = \{N_0, \cdots, N_m\}$ where N_i is the node in T for character $C_i \in \mathcal{C}$. N_0 is the root of T.

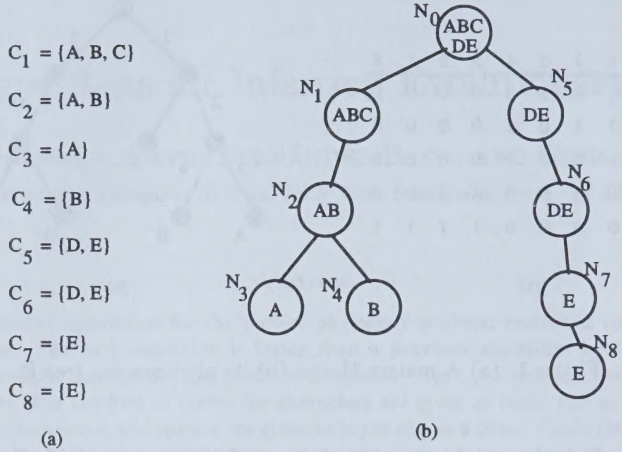


Figure 2: (a) Characters $C_1, \cdots, C_8$ and (b) their character tree T.

(C2) For every $N_r \in V(T)$, if N_r has a parent N_p, then $C_r \subseteq C_p$.

(C3) For every $N_r \in V(T)$, $C_r \cap C_s = \emptyset$, for each sibling N_s of N_r.

By the above lemma, it is clear that $\mathcal{C}$ has a phylogenetic tree P iff it has a character tree. Given a character tree T for $\mathcal{C}$, we can easily construct a phylogenetic tree P for $\mathcal{C}$ as follows. First, for each $r \in \{1, \cdots, m\}$, label the edge between N_r and N_p by r, where N_p is the parent of N_r. Next, consider each N_i in turn. Let N_i be the parent of $N_{i_1}, \cdots, N_{i_r}$, and let $C_R = \cup_{j=1}^{r} C_{i_j}$. If $C_i - C_R = \emptyset$, do nothing. Otherwise, proceed as follows. If N_i is a leaf and $|C_i| = 1$, label N_i with the single species $x \in C_i$. Else, create a child of N_i for each $x \in C_i - C_R$ and label it by x.

Figure 2 shows the characters and the character tree for the example in Figure 1. From now on, we concentrate on constructing T. Algorithm PHYLOGENY, which is described below, creates T (if it exists) from the top down, inserting characters in nonincreasing order of cardinality. Each node has a list of links that point to the children of the node in T. PHYLOGENY also uses an array $A[1..n]$ where $A[i]$ stores the name of the (unique) lowest node in T (constructed so far) containing species i. (Node a is *lower* than node b if the length of the path between a and the root is greater than the length of the path between b and the root.)

Algorithm PHYLOGENY

Step 1. Sort $C_0, \cdots, C_m$ by nonincreasing cardinality. From now on, we assume without loss of generality that $|C_0| \geq |C_1| \geq \cdots \geq |C_m|$. Create a node N_i for each character C_i and initialize all $A[i]$'s to N_0.

Step 2.

```
for $i = 1$ to $m$ do
```

```
begin
    Pick a species $x \in C_i$; let $A[x] = N_l$
    for every $y \in C_i$ do
        if $A[y] \neq A[x]$ then return FAILURE
            else $A[y] \leftarrow N_i$
    Add a link from $N_l$ to $N_i$
end
```

Step 1 can be carried out in $O(n + m + C)$ time by using radix sort with n buckets since for all i, $|C_i| \leq n$. In Step 2, we look at each C_i once and we look at each species in C_i at most once. Thus, Step 2 takes $O(C)$ time. The remaining bookkeeping can also be done in $O(C)$ time. Therefore, the running time of PHYLOGENY is $O(C)$, since usually $n \ll m$ and $m \leq C$.

Lemma 2 *If* PHYLOGENY *returns* FAILURE, *then* $\mathcal{C}$ *has no character tree.*

Proof: Suppose FAILURE occurs when C_i is being considered and we find that $A[y] \neq A[x]$. Let $A[x] = N_j$. Then $C_i \cap C_j \neq \emptyset$ since $x \in C_i \cap C_j$. Since C_j was considered before C_i, $|C_j| \geq |C_i|$. By Lemma 1, $\mathcal{C}$ has no character tree unless $C_j \supseteq C_i$.

Suppose $C_j \supseteq C_i$. We shall show that in this case there exists a C_k such that C_i and C_k violate Lemma 1. Since $A[y] \neq N_j$ and $y \in C_j$, we must have $A[y] = N_k$ where $j < k < i$. Now, $C_i \cap C_k \neq \emptyset$ because $y \in C_i \cap C_k$. Since $A[x] = N_j$ and $k > j$, $x \notin C_k$. Therefore, $x \in C_i$, $x \notin C_k$ and $C_k \not\supseteq C_i$. Since $|C_i| \leq |C_k|$, $C_i \not\supset C_k$ (we use $\subset$ and $\supset$ to denote proper inclusion). Therefore, C_i and C_k violate Lemma 1 and $\mathcal{C}$ has no character tree. □

Lemma 3 *If* PHYLOGENY *succeeds, then the tree constructed is a character tree for* $\mathcal{C}$.

Proof: Since we assume that all characters are nonempty, at the successful termination of PHYLOGENY, all the nodes will be in the constructed tree, say T. Hence T satisfies (C1). We show that conditions (C2) and (C3) are maintained after each iteration of the loop. Initially, these conditions are trivially true. Suppose they hold up to the $(r-1)^{th}$ iteration of the loop where $r > 0$. In the r^{th} iteration, we consider C_r. We argue that conditions (C2) and (C3) hold after the completion of this iteration.

Suppose N_p becomes the parent of N_r and $C_r \not\subseteq C_p$. Then there exists a $y \in C_r$ such that $y \notin C_p$. Thus, $A[y] \neq N_p$. Since N_p is the parent of N_r, the first species x picked from C_r has $A[x] = N_p$. Therefore, $A[y] \neq A[x]$ and PHYLOGENY would return FAILURE. Hence $C_r \subseteq C_p$ and (C2) is satisfied.

If N_r has no siblings, then (C3) trivially holds true. Suppose N_r has siblings and their parent is N_p. Let x be the first species picked from C_r. Since N_p is the parent of N_r, we have $A[x] = N_p$. Let N_s be a sibling of N_r such that $C_r \cap C_s \neq \emptyset$ and let $y \in C_r \cap C_s$. Then $A[y]$ is either N_s or one of the descendants of N_s, but not N_p. Therefore, $A[y] \neq A[x]$ and PHYLOGENY would return FAILURE. Hence $C_r \cap C_s = \emptyset$ and (C3) is satisfied. □

3 An online algorithm for characters

It may sometimes be necessary to update an existing phylogeny or test if a newly introduced character is compatible with the current set of characters. In this section we consider an online algorithm for characters which answers the update and compatibility

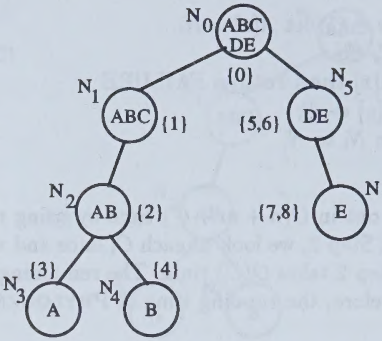


Figure 3: Compact character tree R for T in Figure 2

questions. Note that given a compatible set of characters $\mathcal{C}$, a CT R for $\mathcal{C}$, and a character C_t, the character set $\mathcal{C}' = \mathcal{C} - \{C_t\}$ is also compatible and a CT R' for $\mathcal{C}'$ can easily be constructed from R in $O(|C_t|)$ time. For this reason, in the remainder of this section, we concentrate on the problem of adding new characters.

In the online version for characters, characters arrive one at a time. Without loss of generality, we assume that characters are indexed $C_1, C_2, \cdots$, according to their order of arrival. As before, we stipulate the existence of a character C_0 which contains all n species and that all characters are nonempty. The online binary phylogeny problem is as follows: Given a compatible set of characters $\mathcal{C}_{t-1} = \{C_0, C_1, \cdots, C_{t-1}\}$ a character tree R_{t-1} for $\mathcal{C}_{t-1}$, and a character C_t, determine whether or not $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{C_t\}$ is a compatible set of characters and, if so, construct a character tree R_t for $\mathcal{C}_t$.

For convenience, we will work with a compact version of the character tree, which we shall refer to as a *compact character tree (CCT)*. It is equivalent to a character tree but renders itself more easily to online modifications by identifying nodes with identical characters as a single node. Formally, a CCT for a compatible set of characters $\mathcal{C} = \{C_0, \cdots, C_m\}$ is a tree R with vertex set $V(R) = \{N_0, N_{j_1}, \cdots, N_{j_l}\}$, where $\{j_1, \cdots, j_l\} \subseteq \{1, \cdots, m\}$, together with a labeling function *label* that maps every node in the tree to a subset of $\{1, \cdots, m\}$. R and *label* must satisfy the following properties:

(D1) N_0 is the root for R. (C_0 has all n species.)

(D2) For every $N_t \in V(R)$, if N_t has a parent N_p, then $C_t \subset C_p$.

(D3) For every $N_t \in V(R)$, $C_t \cap C_s = \emptyset$, for each sibling N_s of N_t.

(D4) For every $N_j \in V(R)$, $label(N_j) = \{i : C_i = C_j\}$.

(D5) $\cup\{label(N_j) : N_j \in V(R)\} = \{0, 1, \cdots, m\}$.

Figure 3 shows a CCT for the example in Figure 2. We have the following incremental version of Lemma 1, whose proof is omitted.

Lemma 4 *Suppose $\mathcal{C} = \{C_0, \cdots, C_{t-1}\}$ has a CCT R and C_t is the next character. Then $\mathcal{C} \cup \{C_t\}$ has a CCT R' iff either*

(E1) *$C_t = C_p$ for some $C_p \in \mathcal{C}$, or*

(E2) *There exists $C_p \in \mathcal{C}$ such that $C_t \subset C_p$ and for each child N_r of N_p in R, $C_t \cap C_r = \emptyset$ or $C_t \supset C_r$.*

Note that all characters C_p which satisfy (E2) will be equal and therefore, by (D4) they will be contained in the label set of a unique node in R. Furthermore, if $\mathcal{C} \cup \{C_t\}$ has a CCT R', then it can be obtained from R by carrying out the following transformation:

- If (E1) is satisfied, then add t to $label(N_p)$.
- If (E2) is satisfied, then create a node N_t, assign $label(N_t) = \{t\}$, and make N_p the parent of N_t. The children of N_t are those children N_r of N_p, if any, such that $C_t \supset C_r$.

This transformation is carried out by algorithm ONLINE-PHYLOGENY(R, C_t) given below which assumes that R is not empty (i.e., contains at least the root node N_0) and $C_t \neq \emptyset$. Because of the online requirement, the nodes of the tree will have to carry more information to facilitate fast processing of new characters. Specifically, each node $N_i \in V(R)$ stores (i) the set C_i, denoted by $set(N_i)$, (ii) $|C_i|$, the cardinality of C_i, (iii) $parent(N_i)$, a pointer to the parent of N_i, (iv) $label(N_i)$, the label set of N_i, (v) list of pointers to its children, and (vi) for every $x \in C_i$, a variable $H_i[x]$. $H_i[x]$ is a pointer to the child of N_i whose character contains the species x. If none of these contain x, then $H_i[x]$ is *nil*.

Algorithm ONLINE-PHYLOGENY(R, C_t)

```
     begin
         $R' \leftarrow R$
         Let $v \leftarrow A[l]$ where $l \in C_t$ and $\forall i \in C_t, |set(A[l])| \geq |set(A[i])|$
         while $|C_t| > |set(v)|$ do
             $v \leftarrow parent(v)$
         Let $v = N_p$                                        /* $|C_t| \leq |C_p|$ */
(⋆)      if $C_t \not\subseteq C_p$
(F1)         then return FAILURE
             else
                 if $|C_t| = |C_p|$                              /* $C_t$ is a duplicate of $C_p$ */
(S1)                 then Add $t$ to $label(N_p)$; return $R'$
                     else begin                               /* Now $C_t \subset C_p$ */
                         create a new child $N_t$ for $N_p$
                         $label(N_t) \leftarrow \{t\}$; $parent(N_t) \leftarrow N_p$
                         for every $x \in C_t$ do begin
                             $H_t[x] \leftarrow H_p[x]$; $H_p[x] \leftarrow N_t$
                             if $H_t[x] = nil$
                                 then $A[x] \leftarrow N_t$
                                 else begin
                                     Let $H_t[x] = N_r$
                                     if $parent(N_r) = N_p$
(⋆⋆)                                     then if $C_r \subset C_t$
                                             then begin
```

```
                                Add a link from $N_t$ to $N_r$
                                Remove the link from $N_p$ to $N_r$
                                $parent(N_r) \leftarrow N_t$
                            end
(F2)                        else return FAILURE
                    end
                endfor
(S2)            return $R'$
            end
    end
```

Lemma 5 *If* ONLINE-PHYLOGENY(R, C_t) *returns FAILURE, then* $\mathcal{C} \cup \{C_t\}$ *has no CCT.*

Proof: Suppose FAILURE occurs at (F1) when $|C_t| \leq |C_p|$ but $C_t \not\subseteq C_p$. Then neither one of C_t, C_p contains the other. Since N_p is an ancestor of $A[l]$, $l \in C_p \cap C_t$. Therefore, $C_t \cap C_p \neq \emptyset$ and C_t, C_p violate Lemma 1; thus, $\mathcal{C} \cup \{C_t\}$ has no CCT.

Suppose FAILURE occurs at (F2). Then $C_t \subset C_p$, but there exists $x \in C_t$ such that $H_t[x] = N_r \neq nil$, N_r is a child of N_p, and $C_r \not\subset C_t$. We first show that the node N_p found after the **while** loop is the only candidate for becoming the parent of N_t. Since l must be in the character set of the parent of N_t and $A[l]$ is the lowest node in R containing l, the parent of N_t lies on the path from $A[l]$ to the root of R. No node on the path from $A[l]$ to N_p can be the parent of N_t since the cardinality of its character set is smaller than $|C_t|$. The nodes on the path from N_p to the root of R are not candidates for being a parent of N_t since they have a child which is neither disjoint from nor a subset of C_t. Therefore, N_p is the only candidate for becoming parent of N_t. Now, since $N_r = H_t[x] \neq nil$, $x \in C_r \cap C_t$. Hence, if $C_r \not\subset C_t$, (E2) is violated and $\mathcal{C} \cup \{C_t\}$ has no CCT. □

Lemma 6 *If* ONLINE-PHYLOGENY(R, C_t) *succeeds, then the tree* R' *constructed is a CCT for* $\mathcal{C} \cup \{C_t\}$.

Proof: If R' is returned at (S1), then there exists a $C_p \in \mathcal{C}$ such that $C_p = C_t$. In this case, the algorithm modifies R by adding label t to $label(N_p)$ in accordance with (D4). Since R satisfies (D1)-(D5), R' will also satisfy these conditions.

If R' is returned at (S2), then there is no $C_p \in \mathcal{C}$ such that $C_p = C_t$. In this case, the algorithm adds a new node N_t to R. N_t is made a child of a node N_p such that $C_t \subset C_p$ and made the parent of zero or more nodes N_r such that N_r is a child of N_p in R and $C_r \subset C_t$. Therefore, by (E2) R' is a CCT if C_t is disjoint from all C_r such that N_r is a child of N_p and $C_r \not\subset C_t$. Now if $x \in C_r \cap C_t$, then $x \in set(H_t[x]) \cap C_r$. Since $H_t[x]$ was a sibling of N_r in R, we have a contradiction as R satisfies (D3). □

Running time of ONLINE-PHYLOGENY(R, C_t): The time taken for finding the initial node v is $O(|C_t|)$ since each node N_i stores $|C_i|$. The number of iterations of the **while** loop is at most $|C_t|$ since R is a CCT and by (D2), $|set(parent(v))| \geq |set(v)| + 1$. The only costly operation in the algorithm is checking for set containment in $(\star)$ and $(\star\star)$. Clearly, $(\star)$ is carried out just once. For $(\star\star)$, we have the following. Let $N_{j_1}, \cdots, N_{j_k}$ be the children of N_t. Note that $\sum_{i=1}^{k} |C_{j_i}| \leq |C_t|$. For each child N_r of N_t, the operation $C_r \subset C_t$ is carried out precisely once when $parent(N_r)$ is set to N_t and therefore, for any other $x \in N_t \cap N_r$, the test for $parent(N_r) = N_p$ fails. We shall now present two implementations for carrying out the subset checking and analyze the running time for each case.

Sets as binary search trees: Each node N_i stores the species in C_i as a binary search tree S_i. The time required for checking whether $C_t \subseteq C_p$ is at most $O(|C_t| \log |C_p|)$ since for each element $x \in C_t$, we need to check whether x is in S_p which requires $O(\log |C_p|)$ time. Therefore, algorithm ONLINE-PHYLOGENY(R, C_t) takes $O(|C_t| \log |C_p|)$ time and by invoking it repeatedly, the phylogenetic tree can be found in $O(C \log n)$ time.

Sets as arrays: Each node N_i contains an array S_i of length n where $S_i[x] = 1$ if $x \in C_i$ and $S_i[x] = 0$ otherwise. S_i can be initialized in $O(n)$ time. The node also stores species in C_i as a linked list L_i. The time required to check whether $C_t \subseteq C_p$ is $O(|C_t|)$, since we need to examine each $x \in L_t$ exactly once, to check whether $S_p[x] = 1$. Therefore, algorithm ONLINE-PHYLOGENY(R, C_t) has a running time of $O(n)$ and the array implementation will have a total running time of $O(nm)$ for finding the phylogenetic tree.

4 An online algorithm for species

We now consider what, in a sense, is a dual version of the problem we studied in Section 3, namely, to dynamically maintain a CCT under a sequence of additions and deletions of species. A species S is represented by the set $I(S)$ of the indices of the characters exhibited by S. Given a set $\mathcal{S}$ of species described in this way, there is a corresponding set $\mathcal{C}$ of characters, where each $C_i \in \mathcal{C}$ is given by $C_i = \{S \in \mathcal{S} : i \in I(S)\}$. We shall say that S is *compatible* if the associated set of characters $\mathcal{C}$ is compatible. As before, we shall assume that every species exhibits character C_0; i.e., $0 \in I(S)$ for all $S \in \mathcal{S}$.

Suppose we have a compatible set of species $\mathcal{S} = \{S_1, S_2, \cdots, S_{t-1}\}$ and a CCT R for $\mathcal{S}$. Clearly, for any $S_i \in \mathcal{S}$, the set of species $\mathcal{S}' = \mathcal{S} - \{S_i\}$ is compatible. Constructing a CCT R' for $\mathcal{S}'$ simply requires removing S_i from each node $N_j \in R$ such that $S_i \in C_j$. All of these nodes must lie on a single path in R. The removal of S_i may render identical two nodes which were formerly distinct sets C_j and C_k which differed only in S_i. Also, some set C_j may become empty and thus, N_j may have to be deleted from R. The key to updating R efficiently therefore, lies in representing the nodes and their associated sets properly. For example, if sets are represented as boolean arrays of length n, removal of a species S_i takes $O(1)$ time per set C_j such that $j \in I(S_i)$, for a total of $O(|I(S_i)|)$ time. Using a binary search tree representation of sets, removal of S_i from C_j when $j \in I(S_i)$ takes $O(\log n)$ time, for a total of $O(|I(S_i)| \cdot \log n)$ time per deletion. All other adjustments to R' can be done in $O(|I(S_i)|)$ time. For brevity, we shall leave the details to the reader and turn our attention to the problem of adding a species to $\mathcal{S}$.

Suppose we wish to determine whether $\mathcal{S}' = \mathcal{S} \cup \{S_t\}$ is compatible. Let $I(S_t) = \{0, i_1, i_2, \cdots, i_k\}$ be the indices of the characters of the new species S_t and R be the given tree; i.e., S_t should be added to each of the characters in $\{C_0, C_{i_1}, C_{i_2}, \cdots, C_{i_k}\}$. Our algorithm is based on the following result.

Lemma 7 *Suppose $\mathcal{S} = \{S_1, S_2, \cdots, S_{t-1}\}$ has a CCT R and S_t is the next species with $I(S_t) = \{0, i_1, \cdots, i_k\}$. Let $I_{old} = I(S_t) \cap I(\mathcal{S})$ where $I(\mathcal{S}) = \cup_{S_i \in \mathcal{S}} I(S_i)$. Then $\mathcal{S} \cup \{S_t\}$ is a compatible set of species iff there exists a path P in R from the root to some node $N_l \in V(R)$ such that both of the following conditions hold.*

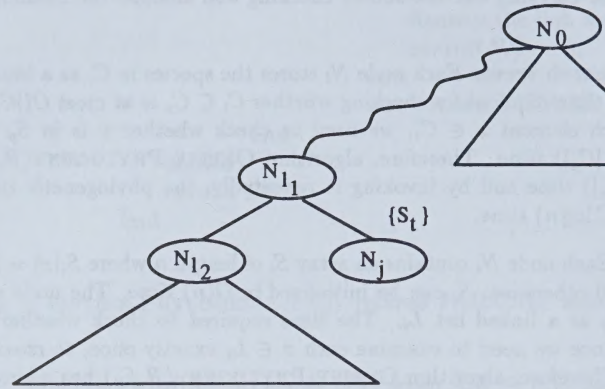


Figure 4: Dividing N_l into N_{l_1} and N_{l_2} and adding N_j

(P1) $I_{old} \subseteq \cup\{label(N_i) : N_i \in P\}$.

(P2) $label(N_l) \cap I(S_t) \neq \emptyset$ *and for each* $N_i \in P - N_l$, $label(N_i) \subseteq I_{old}$.

Proof: We shall first argue that if there is no path P satisfying (P1) and (P2), then $\mathcal{S} \cup \{S_t\}$ is incompatible. Let P' be a longest path satisfying (P2). Note that there always exists such a path P' since $0 \in I(S_t) \cap label(N_0)$ where N_0 is the root of R. Since P' does not satisfy (P1) and every $i \in I(\mathcal{S})$ is present in the label set of exactly one node in R, there exist $N_j \in V(R)$ such that $N_j \notin P'$ and $label(N_j) \cap I(S_t) \neq \emptyset$. Suppose N_j is the highest such node. We now have following cases to consider:

Case 1: Suppose $parent(N_j) \in P'$ and there exists a sibling N_k of N_j such that $N_k \in P'$. By definition, $C_j \cap C_k = \emptyset$. Since $label(N_j) \cap I(S_t) \neq \emptyset$ and $label(N_k) \cap I(S_t) \neq \emptyset$, S_t must be added to both C_j and C_k. Therefore, $C_j \cup \{S_t\}$ and $C_k \cup \{S_t\}$ violate Lemma 1.

Case 2: Suppose $parent(N_j) = N_p \in P'$ and N_j has no sibling that lies on P'. Since P' is of maximum length and $N_j \notin P'$, $label(N_p) \not\subseteq I_{old}$ and $label(N_p) \cap I_{old} \neq \emptyset$. By definition, $C_j \subset C_p$. Let $f \in label(N_p) - I_{old}$. Since $f \notin I(S_t)$ and $j \in I(S_t)$, S_t gets added to C_j but not to C_f. Therefore, $C_j \cup \{S_t\}$ and C_f violate Lemma 1.

Case 3: Suppose $parent(N_j) = N_p \notin P'$. Then $label(N_p) \cap I(S_t) = \emptyset$ and S_t must be added to C_j but not to C_p. But then, $C_j \cup \{S_t\}$ and C_p violate Lemma 1.

In all cases, $\mathcal{S} \cup \{S_t\}$ is incompatible.

Now, suppose there exists a path P satisfying (P1) and (P2). We shall show that, in this case, we can build a CCT R' for $\mathcal{S}' = \mathcal{S} \cup \{S_t\}$ and hence, that $\mathcal{S}'$ is compatible. As before, let N_l be the lowest node in P.

First, let $J_1 = label(N_l) \cap I(S_t)$ and $J_2 = label(N_l) - J_1$. If $J_2 = \emptyset$, then add S_t to every $N_i \in P$. Otherwise, split N_l into two nodes as shown in Figure 4. The top one will be N_{l_1}, where l_1 is any element of J_1 and it will have $label(N_{l_1}) = J_1$. The other node will be N_{l_2}, where l_2 is any element of J_2 and it will have $label(N_{l_2}) = J_2$. Next, if $I_{old} \neq I(S_t)$, then choose any $j \in I(S_t) - I_{old}$ and create a new node N_j, which will be a child of N_{l_1}. Define $label(N_j) = I(S_t) - I_{old}$ and $set(N_j) = \{S_t\}$. Finally, S_t is added to N_{l_1} and to every $N_i \in P - N_l$. It is easy to verify that the resulting tree satisfies (D1)-(D5). □

An algorithm for adding a species to a compatible set of species immediately follows from the proof of the previous lemma. Since there can be only one path P satisfying the conditions of the above lemma, we can find it as follows: First, we find a node $N_l \in V(R)$ such that $label(N_l) \cap I(S_t) \neq \emptyset$ and $|C_l|$ is as small as possible. N_l will be the candidate for the lowest node in the path P. Next, we go up the tree from N_l, testing for each node N_i that we encounter whether $label(N_i) \subseteq I_{old}$. If this does not hold, then we return FAILURE. Otherwise, we test whether the union of the labels we encountered contains I_{old} (note that this can actually be done as we are going up the tree). If not, then return FAILURE. Otherwise, the path P we have found satisfies the conditions of Lemma 7 and we modify the tree as specified in the proof of that lemma.

The lowest node N_l as required above can be efficiently found by keeping an array $B[1..m]$ where $B[i]$ is a pointer to the node $N_j \in V(R)$ such that $i \in label(N_j)$. Most of the work done by the algorithm we have sketched involves testing for a given $N_i \in V(R)$ whether $label(N_i) \subseteq I_{old}$. The only significant problem is the representation of labels so that this test can be done efficiently. As done for $set(N_i)$ in Section 3, we can implement $label(N_i)$ as either a boolean array of length m or as a binary search tree. The array implementation leads to a $O(m)$ running time per update and to a $O(nm)$ algorithm for adding n species. The binary search tree implementation leads to a $O(|I(S_t)| \cdot \log m)$ running time per update and to a $O(C \log m)$ running time algorithm for adding n species.

References

[1] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, Springer Verlag, Lecture Notes in Computer Science, to appear, 1992.

[2] G. F. Estabrook, C. S. Johnson Jr., and F. R. McMorris. An idealized concept of the true cladistic character. *Mathematical Biosciences*, 23, 263–272, 1975.

[3] G. F. Estabrook. Cladistic methodology: A discussion of the theoretical basis for the induction of evolutionary history. *Annual Review of Ecology and Systematics*, Vol. 3, 427–456, 1972.

[4] W. M. Fitch. Aspects of Molecular Evolution. *Annual Reviews of Genetics*, Vol. 7, 343–380, 1973.

[5] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, Vol. 21, 19–28, 1991.

Existence and Design of Supervisors for Vector Discrete Event Systems

S. L. Chen W. M. Wonham
chensl@odin.control.utoronto.ca wonham@odin.control.utoronto.ca
Systems Control Group
Electrical Engineering Department
University of Toronto
Toronto, Ont.
Canada M5S 1A4

Abstract

A Vector Discrete Event System (VDES) is a Discrete Event System in which the state space and transition functions have arithmetic additive structure. The paper studies supervisory control of VDES, with emphasis on closed-form solutions (CFS) for supervisors. Sufficient conditions for the existence of CFS as well as a systematic approach to synthesize CFS are provided. A software tool, VTCT, for analysis and synthesis of VDES is introduced.

1 Introduction

A Vector Discrete Event System (VDES) is a Discrete Event System (DES) with state space and transition function comprising a vector addition system. Supervisory control of VDES has been studied by Huang, Li and Wonham[2][3][4][5][6][7][8]. The objective of control is to find an (optimal) control law such that the controlled behaviour of the system is, in the sense of set inclusion, the largest controllable sublanguage that satisfies a legality specification. In this context, a significant question is the existence of a controller which itself is of VDES type, i.e. the so-called Closed-Form Solution (CFS). Sufficient conditions for the existence of a closed-form solution have been provided by Huang and Li[2][8]. These conditions overlap one another in scope; some structural properties that must be satisfied for Huang's theorem [2] are not required for Li's theorems [8] and vice versa. This suggests that a new existence theorem should be sought which contains the results of Huang and Li as special cases. Such a theorem, together with a construction procedure for the CFS whose existence is asserted by the theorem, is introduced in this paper. Also considered is modular supervisory control of VDES, relevant to the study of the existence of CFS.

2 Introduction to VDES Models

A Vector Discrete Event System is formally defined by a 4-tuple [8]:

$$G = (\mathbf{X},\ \Sigma,\ \delta,\ X_0)$$

where $\mathbf{X} = \mathbf{Z^n} = \{[x_1, x_2, \cdots, x_n] \mid x_i \in \mathbf{Z}\}$ is the state space; X_0 is the (nonnegative) initial state; Σ is a finite event set; Σ is partitioned into two subsets, the controllable event set Σ_c and uncontrollable event set Σ_u. Each event in Σ_c can be disabled by a controller while no event in Σ_u can be disabled. $\delta : \Sigma \times \mathbf{X} \longrightarrow \mathbf{X}$ is the partial transition function. Given an event $\alpha \in \Sigma$ and $X \in \mathbf{X}$, the transition function is defined, denoted $\delta(\alpha, X)!$, iff $X + E_\alpha \geq 0$. Also,

$$\delta(\alpha, X)! \Rightarrow \delta(\alpha, X) := X + E_\alpha.$$

Here E_α is the *displacement vector* of the event α. The function δ can be extended to $\Sigma^* \times \mathbf{X}$ by

$$\delta(\epsilon, X) := X \qquad \delta(s\alpha, X) := \delta(\alpha, \delta(s, X))$$

The behaviour of a VDES G is described by the *language* generated by the VDES, denoted by L(G), and defined as:

$$L(G) = \{\ s \in \Sigma^* \ \mid\ \delta(s, X_0)!\}$$

Definition Given a VDES G=$(\mathbf{X}, \Sigma, \delta, X_0)$ and a VDES E= $(\mathbf{Y}, \Sigma, \xi, Y_0)$, E_α and H_α denote the displacement vector of α in VDES G and VDES E respectively. The *synchronous product* of E and G is a VDES

$$G||_s E = (\mathbf{X} \oplus \mathbf{Y}, \Sigma, \delta \times \xi, X_0 \oplus Y_0)$$

Here the transition function is defined as:

$$(\delta \times \xi)(\alpha, X \oplus Y)! \Leftrightarrow \delta(\alpha, X)! \bigwedge \xi(\alpha, Y)!$$

and then

$$(\delta \times \xi)(\alpha, X \oplus Y) := \delta(\alpha, X) \oplus \xi(\alpha, Y)$$

□

Theorem L(G$||_s$E) = L(G) ∩ L(E)

Proof: Proposition 3.2 of [2]. □

'Loop-freedom' or the property of being 'loop-free' is a structural property of VDES which is important to the later discussion. In [2], a loop is defined as follows.

Definition A VDES G=$(\mathbf{X}, \Sigma, \delta, X_0)$ has a *loop* w.r.t. $\Sigma' \subseteq \Sigma$ if there exist $\alpha_1, \alpha_2, \cdots, \alpha_p \in \Sigma'$ and state variables $x_1, x_2, \cdots, x_p$ s.t.

$$E_{\alpha_1}(x_1) < 0,\ E_{\alpha_2}(x_2) < 0,\ \cdots,\ E_{\alpha_p}(x_p) < 0$$
$$E_{\alpha_1}(x_2) > 0,\ E_{\alpha_2}(x_3) > 0,\ \cdots,\ E_{\alpha_p}(x_1) > 0$$

A VDES is *loop-free* w.r.t. Σ' iff it has no loop w.r.t. Σ'. □

3 Supervisory Control of VDES and Modularity

Given a plant VDES G and a specification VDES E, E (or L(E)) is *controllable* w.r.t. L(G) iff $\overline{L(E)}\Sigma_u \cap L(G) \subseteq \overline{L(E)}$.

Define $C(E) = \{ K \subseteq L(E) \cap L(G) \mid K$ is controllable w.r.t. $L(G) \}$. The supremal element of $C(E)$ is denoted by $supC(E)$.

Also define $\Gamma = \{ \gamma \mid \gamma \in 2^{\Sigma}, \Sigma_u \subseteq \gamma \}$. Γ is called the ***admissible control set*** of G or the *set of control patterns*. Every element of Γ is a set of events to be enabled.

Definition A supervisory control of a VDES G is a map $V : L(G) \rightarrow \Gamma$. The language of the VDES G under the control V, denoted by L(V/G), is defined inductively as:
(1) $\epsilon \in L(V/G)$.
(2) For $s \in L(V/G)$, if $\sigma \in V(s)$ and $s\sigma \in L(G)$ then $s\sigma \in L(V/G)$.
(3) No other string belongs to L(V/G).

The objective of control is to a find control law V^* s.t. $L(V^*/G) = supC(E)$ for a given specification VDES E.

Theorem 1 Given a plant VDES G and specification VDES E, we have [2]

$$supC(E) = \{t \in L(E) \cap L(G) \mid (\forall s \leq t)(\forall l \in \Sigma_u^*)\ sl \in L(G) \Rightarrow sl \in L(E)\}$$

□

Practical problems may be complicated enough (say, the specification VDES E is high dimensional) to make the design of a centralized control law difficult. A question that naturally arises is: is it possible to divide the overall control task into several subtasks each of which can be solved more easily, and when all subcontrollers run concurrently the overall control task is performed?

Decomposition Theorem Given an n-dimensional VDES E, there exists an (n-1)-dimensional VDES E_1 and a one-dimensional VDES E_2, s.t. $E = E_1||_s E_2$ and $L(E) = L(E_1) \cap L(E_2)$. □

Corollary Given an n-dimensional VDES E, there exist n one-dimensional VDES E_i (i=1,2,$\cdots$,n) s.t.

$$E = E_1||_s E_2||_s \cdots ||_s E_n$$

and

$$L(E) = \bigcap_{i=1}^{n} L(E_i)$$

□

The last corollary answers the question of how to divide the overall control task into subtasks.

Theorem Given a plant VDES G and a specification VDES E, if there exist a VDES E_1 and a VDES E_2 s.t. $L(E_1) \cap L(E_2) = L(E)$, then

$$supC(E) = supC(E_1) \cap supC(E_2)$$

□

The last theorem and corollary indicate that whenever the specification is a VDES, the modular approach can be invoked for the synthesis of the supervisor.

4 Existence of a Closed-Form Solution

An important issue in the context of supervisory control of VDES is the existence of a controller which is itself of VDES type (closed-form solution). With the existence of the CFS, controlled behaviour of plants can be studied by 'off-line' computation instead of 'on-line' simulation. 'On-line' simulation may be problematic since at each step of simulation certain linear integer programming problems [7] have to be solved. Also, the existence of a CFS is advantageous to hardware implementation of controllers.

Given a plant VDES G and a one-dimensional specification VDES E, let E_α and H_α denote the displacement vector of α in G and E respectively. We define

$$\overline{S} = \{\ \alpha \ \mid \ \alpha \in \Sigma_u,\ H_\alpha < 0\ \}$$

$$\begin{array}{ll} \alpha^- = \{x \ \mid \ E_\alpha(x) < 0\} & \alpha^+ = \{x \ \mid \ E_\alpha(x) > 0\} \\ \hat{\alpha}^- = \{y \ \mid \ H_\alpha(y) < 0\} & \hat{\alpha}^+ = \{y \ \mid \ H_\alpha(y) > 0\} \\ x^- = \{\alpha \ \mid \ E_\alpha(x) > 0\} & x^+ = \{\alpha \ \mid \ E_\alpha(x) < 0\} \\ x^\Sigma = x^+ \cup x^- & \end{array}$$

Definition Given a plant VDES G and a one-dimensional VDES E, the *subsystem induced by the specification* , denoted by $(\Sigma'(\overline{S}), \Omega'(\overline{S}))$, is defined inductively as follows:
(1) $\overline{S} \subseteq \Sigma'(\overline{S})$; $\ \cup_{\alpha \in \overline{S}}\, \alpha^- \subseteq \Omega'(\overline{S})$;
(2) $(\alpha_i \in \Sigma'(\overline{S})) \bigwedge (x_k \in \alpha_i^-) \Rightarrow (x_k \in \Omega'(\overline{S}))$;
(3) $(\alpha_i \in \Sigma'(\overline{S})) \bigwedge (\alpha_j \in (x_k^- \cap \Sigma_u)) \bigwedge (x_k \in \alpha_i^-) \Rightarrow (\alpha_j \in \Sigma'(\overline{S}))$;
(4) All the elements of $\Sigma'(\overline{S})$ and $\Omega'(\overline{S})$are obtained as in (1), (2) and (3). □

Definition The *displacement vector* of $\alpha \in \Sigma'(\overline{S})$ is defined as :

$$E^s_\alpha(x) = \begin{cases} 0 & x \notin \Omega'(\overline{S}) \\ E_\alpha(x) & x \in \Omega'(\overline{S}) \end{cases}$$

□

Definition Given a plant VDES G and a specification VDES E, if there exists a VDES CFS s.t. $L(CFS) \cap L(G) = supC(E)$, then the VDES CFS is called a *closed-form solution* for the optimal supervisory control. □

Existence Theorem Given a plant VDES G and a one-dimensional specification VDES E. If the plant satisfies:

(1) $(\Sigma'(\overline{S}), \Omega'(\overline{S}))$ is loop-free w.r.t. Σ_u;
(2) $\forall \alpha \in \Sigma_u$, E_α has at most one negative component, which is -1;
then a closed-form solution VDES CFS exists. □

Under the assumption that the conditions of the Existence Theorem are satisfied, an operator defined in [2] is adapted to calculate the VDES CFS.

Definition Under the assumption that the plant VDES G and the one-dimensional specification VDES E satisfy the conditions of Existence Theorem, for those $\alpha \in \Sigma_u$ with $\hat{\alpha}^- \neq \emptyset$ and $\alpha^- = \{x\} \neq \emptyset$, where x is a state variable, an operator

$$T_\alpha : VDES \longrightarrow VDES$$

is defined as follows:

$$\mathrm{E_{new}} = T_\alpha(\mathrm{E}) := (\mathbf{Y}, \Sigma, \xi_{new}, Y_{0,new})$$

The new initial state $Y_{0,new}$ and the new displacement vector $H_{\sigma,new}$ for $\sigma \in \Sigma$ are defined as:
(1) $Y_{0,new} = Y_0 + X_0(x) H_\alpha$.
(2) $(\forall \sigma \in \Sigma)$ $H_{\sigma,new} = H_\sigma + E_\sigma(x) H_\alpha$. □

Given a plant VDES G and a specification VDES E, if L(E) is controllable w.r.t. L(G), then $supC(\mathrm{E}) = \mathrm{L(G)} \cap \mathrm{L(E)}$, i.e. the specification itself is the closed-form solution. It can be easily proved that, for the given specification VDES E, if $\forall \alpha \in \Sigma_u$, $H_\alpha \geq 0$, then L(E) is controllable w.r.t. L(G). A simple calculation shows that, if $\mathrm{E_{new}} = T_\alpha(\mathrm{E})$, then $H_{\alpha,new} = 0$; also, if E is one-dimensional, then $\mathrm{E_{new}}$ is one dimensional.

The following lemmas present the property of the T_α operator.

Lemma 1 If $\mathrm{E_{new}} = T_\alpha(\mathrm{E})$, then

$$\mathrm{L(G)} \cap \mathrm{L(E_{new})} \subseteq \mathrm{L(E)} \cap \mathrm{L(G)}$$

Proof: Lemma 4.11 of [1]. □

Lemma 2 Let $s \in \mathrm{L(G)} \cap \mathrm{L(E_{new})}$. Then

$$(\forall l \in \Sigma_u^*)\ sl \in \mathrm{L(G)} \Rightarrow sl \in \mathrm{L(E)}$$

iff

$$(\forall l \in \Sigma_u^*)\ sl \in \mathrm{L(G)} \Rightarrow sl \in \mathrm{L(E_{new})}$$

Proof: Lemma 4.12 of [1]. □

Lemma 3 $supC(\mathrm{E_{new}}) = supC(\mathrm{E})$
Proof: This can be proved by induction on length of strings s.
If $s = \epsilon$ then the result is trivial, i.e.

$$\epsilon \in supC(\mathrm{E}) \Leftrightarrow \epsilon \in supC(\mathrm{E_{new}})$$

Assume that for any s with $|s| \leq k$ $(k \geq 1)$

$$s \in supC(\mathrm{E}) \Leftrightarrow s \in supC(\mathrm{E_{new}})$$

In the case of $|s| = k+1$:
$s \in supC(\mathrm{E}) \Rightarrow (s \in \mathrm{L(E)} \cap \mathrm{L(G)}) \bigwedge (s \in \mathrm{L(E_{new})} \cap \mathrm{L(G)})$, i.e. $s \in \mathrm{L(E)} \cap \mathrm{L(E_{new})}$.
By Lemma 3 and Theorem 1, $s \in supC(\mathrm{E_{new}})$
Similarly $s \in supC(\mathrm{E_{new}}) \Rightarrow s \in \mathrm{L(E)} \cap \mathrm{L(E_{new})}$
So by Lemma 3 and Theorem 1, $s \in supC(\mathrm{E})$, i.e. $s \in supC(\mathrm{E_{new}}) \Leftrightarrow s \in supC(\mathrm{E})$. i.e. $supC(\mathrm{E}) = supC(\mathrm{E_{new}})$. ☐

Definition Given a plant VDES G, let $\Sigma' \subset \Sigma$, $\alpha \in \Sigma$. Then α is a *leaf event* w.r.t. Σ' iff

$$(\forall x_i \in \alpha^+) \;\; x_i^+ \cap \Sigma' = \emptyset$$

i.e. $(\alpha^+)^+ \cap \Sigma' = \emptyset$. ☐

In the imagery of Petri nets, a leaf event α w.r.t. Σ' is an event that does not deposit a token in any place or, if it deposits a token in a place x, no other event in Σ' will remove the token from x.

Given a plant VDES G and a one-dimensional VDES E, if the conditions of the Existence Theorem are satisfied, the following procedure calculates the closed-form solution.

```
Procedure;
begin
    Compute $(\Sigma'(\overline{S}), \Omega'(\overline{S}))$;
    Check conditions of the Existence Theorem:  if failure then exit;
    $k := |\Sigma'(\overline{S})|$;
    for $i := 1$ to $k$ do
        begin
            find leaf event $\alpha_i$ w.r.t. $\Sigma'(\overline{S})$;
            $E := T_{\alpha_i}(E)$;
            $\Sigma'(\overline{S}) := \Sigma'(\overline{S}) - \{\alpha_i\}$;
        end ;
    VDES CFS:=E;
end.
```

Proposition The VDES CFS obtained by using Procedure is the desired closed-form solution. ☐

By use of modular supervisory control, the Existence Theorem can be extended to the situation where the given specification is an n-dimensional VDES E.

Corollary Given a plant VDES G and an n-dimensional specification VDES E s.t. there exist n one-dimensional VDES $\mathrm{E_i}$ (i=1,2,$\cdots$,n) satisfying

$$\mathrm{E} = \mathrm{E_1} ||_s \mathrm{E_2} ||_s \cdots ||_s \mathrm{E_n}$$

If for each E_i, the plant VDES G satisfies the conditions of the Existence Theorem, then the closed-form solution exists. □

For a plant VDES G and a specification VDES E satisfying the Corollary, we can use Procedure to calculate VDES CFS_i for the VDES E_i. The overall closed-form solution is

$$CFS = CFS_1 ||_s CFS_2 ||_s \cdots ||_s CFS_n$$

Corollary Given a plant VDES G and a one-dimensional specification VDES E, condition (2) of the Existence Theorem can be weakened as follows:
(2′) $\forall \alpha \in \Sigma'(\overline{S})$, E_α has at most one negative component, which is -1. □

Similarly, the last corollary can be extended to the situation that the specification is an n-dimensional VDES E.

Let n_u be the number of uncontrollable events; dim_G be the dimension of the VDES G; dim_E be the dimension of VDES E. Then the computational complexity of calculating VDES CFS is $O(dim_E(n_u^3 dim_G))$.

All the results obtained so far have been implemented in a software package VTCT, which thus provides an analysis and synthesis tool for VDES.

5 An Example: Manufacturing Cell

Consider a manufacturing cell consisting of a main machine and an auxiliary machine and a test unit. Each part should be processed sequentially by the main machine and the auxiliary machine. Then the processed part is checked by the test unit. If the part is up to standard (i.e. acceptable) then it will be sent to other cells for processing; otherwise it will be discarded as defective. The main machine can be restarted if necessary.

The specifications for the operation of the cell are:

(1) Each part should be processed exactly three times. The part is discarded iff the part is unacceptable after the third processing.

(2) If three parts have been discarded then the main machine should be restarted.

(3) The main machine cannot be restarted unless the auxiliary machine is idle or off-line.

(4) The auxiliary machine should be on-line before the main machine is ready for operation.

(5) The cell can process at most one part at each time.

(6) The buffer of the test unit with capacity 1 must not overflow.

The block diagram of the cell is shown in Fig. 1. VDES model of the cell is shown in Fig. 2. Events a_i are controllable and events b_i are uncontrollable. VDES models for each specification are shown in Fig. 3.

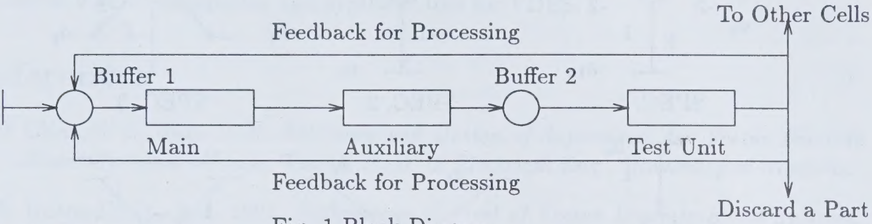

To Other Cells
Feedback for Processing
Buffer 1
Buffer 2
Main
Auxiliary
Test Unit
Feedback for Processing
Discard a Part

Fig. 1 Block Diagram

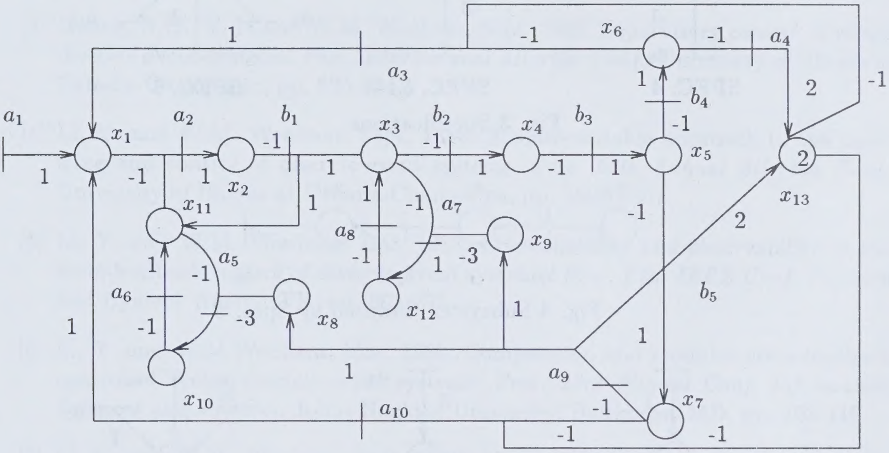

a_1
x_1
a_2
x_2
b_1
x_3
b_2
x_4
b_3
x_5
x_6
a_4
a_3
b_4
x_{13}
2
x_{11}
a_5
a_6
x_{10}
x_8
a_7
a_8
x_9
x_{12}
b_5
a_9
x_7
a_{10}
1
-1
2
-3

Fig. 2 VDES Model

a_1: delivery of an unprocessed part to the main machine;

a_2: the main machine starts working; a_9: discard an unacceptable part;

a_3, a_{10}: feedback of a part for reprocessing;

a_4: delivery of an acceptable part to other calls;

a_5: restart the main machine; b_1: the main machine finishes working;

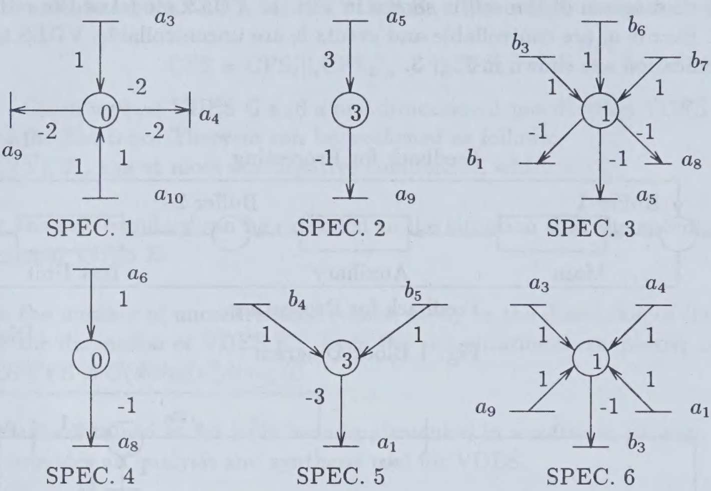
a_6: the main machine becomes ready for operation;

a_7 (a_8): the auxiliary machine goes off-line (on-line);

b_2 (b_3): the auxiliary machine starts (finishes) working;

b_4 (b_5): a processed part is tested to be acceptable (unacceptable);

Controllers for each specification can be obtained via the procedure introduced in the paper (Fig. 5). For example, for specification 6, the subsystem induced by the specification is shown in Fig. 4 and it can be easily checked that all the conditions of the Existence Theorem are satisfied. The controllers shown in Fig. 5 were computed using VTCT.


a_3
1
-2
0
-2
a_4
-2
a_9
1
1
a_{10}
a_5
3
3
-1
a_9
b_6
b_3
1
b_7
1
1
1
-1
-1
b_1
-1
a_8
a_5
a_6
1
0
-1
a_8
b_4
b_5
1
1
3
-3
a_1
a_3
a_4
1
1
1
1
1
a_9
-1
a_{10}
b_3

SPEC. 1 SPEC. 2 SPEC. 3

SPEC. 4 SPEC. 5 SPEC. 6

Fig. 3 Specifications

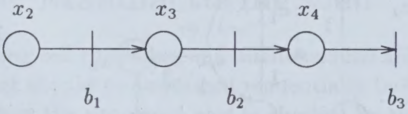
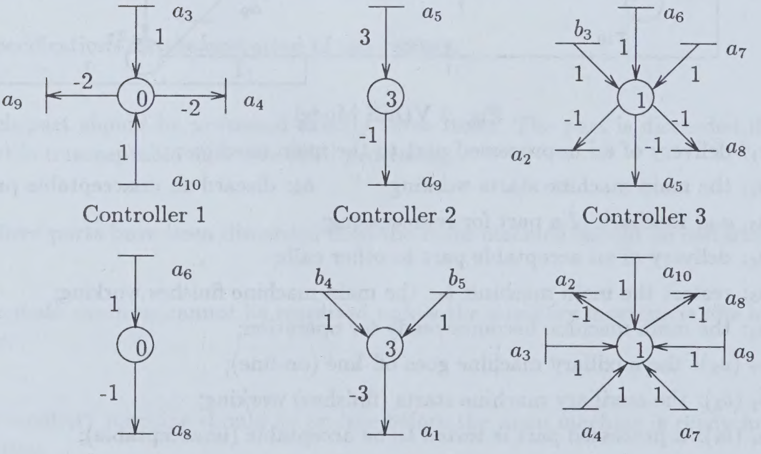

x_2
x_3
x_4
b_1
b_2
b_3

Fig. 4 Subsystem Induced by Spec. 6


a_3
1
-2
a_9
0
-2
a_4
1
a_{10}
a_5
3
3
-1
a_9
a_6
b_3
1
a_7
1
1
1
-1
-1
a_2
-1
a_8
a_5
a_6
1
0
-1
a_8
b_4
b_5
1
1
3
-3
a_1
a_2
1
a_{10}
a_8
-1
-1
a_3
1
1
a_9
1
1
1
a_4
a_7

Controller 1 Controller 2 Controller 3

Controller 4 Controller 5 Controller 6

Fig. 5 Controllers

6 Conclusions

The paper studies supervisory control of VDES. In this context, a significant question is the existence of a controller which itself is of VDES type, i.e. the so-called closed-form solution for controllers. Weaker sufficient conditions for the existence of a closed-form solution than were known previously are provided in the paper. A systematic approach is developed to calculate closed-form solutions whose existence is guaranteed by our sufficient conditions. Procedures are implemented to compute the closed-form solutions as part of VTCT, an analysis and synthesis tool for VDES.

References

[1] Chen, S. L., July. 1992. *Existence and Design of Supervisors for Vector Discrete Event Systems.* M.A.Sc. Thesis, Dept. of Electrical Eng., University of Toronto.

[2] Huang, N.Q., Apr. 1991. *Supervisory Control of Vector Discrete-Event Systems.* M.A.Sc. Thesis, Dept. of Electrical Eng., University of Toronto.

[3] Huang, N.Q., Y. Li and W.M. Wonham, Sept. 1989. Supervisory control of vector discrete event systems. *Proc. 27th Annual Allerton Conf.*, University of Illinois at Urbana-Champaign, pp. 925-934.

[4] Li, Y. and W.M. Wonham, Sept. 1988. A state-variable approach to the modeling and control of discrete-event systems. *Proc. 26th Annual Allerton Conf.*, University of Illinois at Urbana-Champaign, pp. 1502-1507.

[5] Li, Y. and W.M. Wonham, Dec. 1988. Controllability and observability in the state-feedback control of discrete-event systems. *Proc. 27th IEEE Conf. Decision and Control*, Austin, TX, pp. 203-208.

[6] Li, Y. and W.M.Wonham, Mar. 1989. Composition and modular state-feedback control of vector discrete-event systems. *Proc. 23rd Annual Conf. Information Sciences and Systems*, Johns Hopkins University, Baltimore, MD, pp. 103-110.

[7] Li, Y. and W.M. Wonham, Sept. 1989. Linear integer programming techniques in the control of vector discrete-event systems. *Proc. 27th Annual Allerton Conf*, University of Illinois at Urbana-Champaign, pp. 528-537.

[8] Li.Y., June 1991. *Control of Vector Discrete-Event Systems.* Ph.D. Thesis, Dept. of Electrical Eng., University of Toronto.

[9] Peterson, J., 1981. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ.

Hierarchical and Modular Control of Discrete-Event Systems

K. C. Wong and W. M. Wonham

Systems Control Group
Dept. of Electrical Engineering
University of Toronto
Toronto, CANADA M5S 1A4

Abstract

In this paper we formalize the concepts of hierarchical consistency with respect to nonblocking and coordination on the basis of an abstract notion of control structure which is independent of any specific control technology. The concepts of observer and of postfix closure play an important role in the development. Provided the markings in the two levels of a hierarchy are 'consistent' and the aggregation map is strongly observable, the nonblocking property is preserved between the two levels. Under similar conditions conflicts among modular supervisors can be resolved by a hierarchical-coordination scheme.

1 Introduction

Architectural decomposition(horizontal and vertical modularity) is a familiar feature in the analysis, design and control of complex dynamical systems. In this paper, we study these architectures in the setting of discrete-event systems (DES) - systems or processes that are discrete in time and state space, asynchronous or event driven, and possibly nondeterministic. Examples include manufacturing systems, traffic systems, database management systems, communication protocols and logistic systems. A control theory (RW) for a general class of DESs was initiated by Ramadge and Wonham (for a review see [RW89]). Control-theoretic problems such as synthesis of controlled dynamic invariants by feedback and concepts such as controllability and observability have been investigated within this framework. For computation in the RW framework, the specifications, plants and controllers must (currently) be presented as finite transition structures. Then the effort of computing the controller given the plant and specification is of polynomial order in the model's state size. However, the plant is often a product of a number of simple components, and thus its state size increases exponentially with the number of components. As pointed out in [Won91], one strategy to contend with the combinatorial complexity of DESs is architectural decomposition, via concepts of modular, decentralized and hierarchical control. Within the RW framework, these architectural concepts have been investigated by a number of workers in the field [RW88], [LW88], [RW91], and [ZW90]. The need for a unified development of these concepts is pointed out in the conclusion of [ZW90]. Recently, in order to have greater modelling capacity and realism, additional features such as time have been introduced into the RW framework. A control theory of timed discrete-event systems (TDESs) with its own "control technology" - the specific manner in which control is exercised - is developed in the spirit of the standard RW theory in [BW92]. With time being explicitly modelled, the combinatorial complexity of TDESs is inevitably more problematic.

We shall formalize some concepts of control architecture, especially that of hierarchical consistency and coordination, in a unified, algebraic manner in the linguistic setting. The main focus of this study is the property of nonblocking, introduced in the original RW theory. The development is based on an abstract notion of control, which is inspired by the standard control technology in the RW theory but is independent of any specific technologies, so that the theory can be adapted to different extensions of the RW theory such as the control theory of TDESs [BW92]. It is found that a notion of observability together with control and the concept of postfix closure play an essential role in the development.

The paper is organized as follows. In section 2, we formalize the hierarchical control structure, examine a general notion of hierarchical consistency, and investigate how the nonblocking property is preserved between adjacent hierarchical levels. In section 3, we formalize a notion of hierarchical coordination for resolving conflicts among modular supervisors.

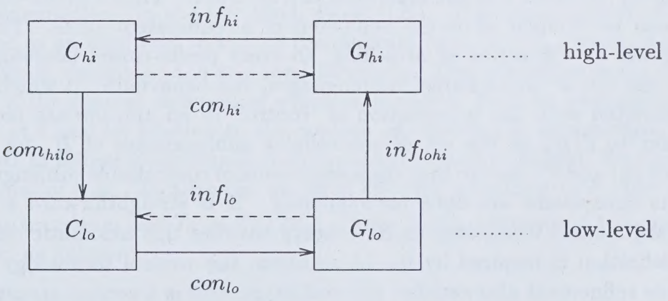
2 Hierarchical Control

Let Σ be an alphabet representing the events in the processes, Σ^+ be the set of all finite strings with events in the alphabet Σ, and $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where $\epsilon \notin \Sigma$ denotes the empty string. A string $s' \in \Sigma^*$ is a *prefix* of $s \in \Sigma^*$, denoted $s' \leq s$, if there exists $u \in \Sigma^*$ such that $s'u = s$. The *prefix closure* of a language $H \subseteq \Sigma^*$, denoted $\overline{H}$, is the set of all strings in Σ^* that are prefixes of the strings in H, i.e.,

$$\overline{H} = \{s \in \Sigma^* \,|\, (\exists u \in H)\, s \leq u\}$$

A language H is *prefix-closed* if $\overline{H} = H$. We denote the power set of a language $L \subseteq \Sigma^*$ by $\mathcal{P}(L)$. Let $\mathcal{F}_L := \{H \in \mathcal{P}(L) \,|\, H \text{ is prefix-closed }\}$ for a prefix-closed language $L \subseteq \Sigma^*$. It is easy to verify that $\mathcal{F}_L$ is a complete sublattice of $\mathcal{P}(L)$. The prefix closure can be also be viewed as an operator on $\mathcal{P}(\Sigma^*)$. For notational convenience, we may denote the prefix closure of H as $pre(H)$. The prefix-closed languages are the fixpoints of pre. For a prefix-closed language $L \subseteq \Sigma^*$, $\mathcal{P}(L)$ is closed under pre, so we can write $pre : \mathcal{P}(L) \longrightarrow \mathcal{P}(L)$. Let $Func(L) := \{f \,|\, f : \mathcal{P}(L) \longrightarrow \mathcal{P}(L)\}$ for a prefix-closed language $L \subseteq \Sigma^*$. So $pre \in Func(L)$. The natural partial order $\subseteq$ on $\mathcal{P}(\Sigma^*)$ restricted to $\mathcal{P}(L)$ induces a partial order $\leq$ on $Func(L)$ as follows: let $f, g \in Func(L)$. Then $f \leq g$ if for all $H \in \mathcal{P}(L)$ $f(H) \subseteq g(H)$. We denote the composition of functions by $\circ$.

Following [ZW90], we consider a two-level hierarchy as shown in the following.



Here G_{lo} is the actual plant to be controlled by C_{lo}; while G_{hi} is an abstract, simplified model of G_{lo}. Suppose G_{hi} is also endowed with control structure. Then C_{hi} could in principle exercise control over G_{hi}. But, in fact, this control of C_{hi} is only virtual, in that G_{hi} is entirely driven by G_{lo} via the information inf_{lohi}. As a result, the high-level control is realized by a sequence of actions: first, the intention of C_{hi} is conveyed to C_{lo} through the command channel com_{hilo}. Then the control is carried out by con_{lo}. Finally, the results of the control action are summarized and reported to C_{hi} via inf_{lohi} and inf_{hi}. Now let Σ and T be the alphabets representing the events of the low and high level respectively. Suppose $L_m \subseteq L \subseteq \Sigma^*$ given, with L being prefix-closed. Here L represents the behaviour of G_{lo} and L_m is the set of 'marked' or significant strings. From now on, we assume that the map $\theta : L \longrightarrow T^*$ is *prefix-preserving* (i.e., past-preserving or causal), namely the following diagram commutes.

$$\begin{tikzcd}
\mathcal{P}(T^*) \arrow[r, "pre"] & \mathcal{P}(T^*) \\
\mathcal{P}(L) \arrow[r, "pre"] \arrow[u, "\theta"] & \mathcal{P}(L) \arrow[u, "\theta"']
\end{tikzcd}$$

This definition of θ is equivalent to that in [ZW90]. Now let $M_m = \theta(L_m)$ and $M = \theta(L)$ be the marked and closed behaviours of G_{hi} respectively. Then the inverse image map $\theta^{-1} : \mathcal{P}(M) \longrightarrow \mathcal{P}(L)$ represents the information channel which conveys the high-level commands to the low level. A natural question to investigate is whether θ^{-1} commutes with pre, i.e.,

$$\theta^{-1} \circ pre \overset{?}{=} pre \circ \theta^{-1}$$

It turns out that $pre \circ \theta^{-1} \leq \theta^{-1} \circ pre$ always holds, whereas the reverse inequality is problematic. The equality relates to an observer property which will be crucial later. Next we introduce abstractly a control structure on L. The map $\mathcal{C} : \mathcal{F}_L \longrightarrow \mathcal{P}^2(L)(= \mathcal{P}\mathcal{P}(L))$ is a *control structure on* L if

(1) For every $H \in \mathcal{F}_L$, $\mathcal{C}(H) \subseteq \mathcal{P}(H)$ is a complete upper semilattice with respect to $\cup$ in $\mathcal{P}(H)$
(2) For every $H \in \mathcal{F}_L$, $\emptyset, H \in \mathcal{C}(H)$
(3) $K \in \mathcal{C}(H) \Longrightarrow \overline{K} \in \mathcal{C}(H)$
(4) For every $H, F \in \mathcal{F}_L$ with $H \subseteq F$, $\mathcal{C}(F) \cap \mathcal{P}(H) \subseteq \mathcal{C}(H)$
(5) For every $F \in \mathcal{F}_L$ and every $H \in \mathcal{F}_F \cap \mathcal{C}(F)$, $\mathcal{C}(F) \cap \mathcal{P}(H) = \mathcal{C}(H)$

Intuitively, consider L as the behaviour of a system G. Then a prefix-closed sublanguage of L can be thought of as the behaviour of a 'subsystem' of G. The map $\mathcal{C}$ can be thought of as the action of attaching, to every prefix-closed language or 'subsystem' of G, the set of 'controllable' sublanguages, the behaviours to which the system can be restricted with the intervention of 'control' in an appropriate technology. Hence we refer to $\mathcal{C}(H)$ as the set of controllable sublanguages of H, for every $H \in \mathcal{F}_L$. Axioms (4) and (5) ensure that the assignments of controllable sublanguages of a system and its subsystems are done harmoniously. It is straightforward to verify that the standard control technology in RW theory satisfies this axiomatic definition. In fact, this definition is inspired by it. In addition, the control technology in [BW92] with suitable refinement also satisfies this definition, and is a control structure in our sense. Further restrictions on the control structure will be brought in as they are needed. Next we let $\kappa_F : \mathcal{P}(F) \longrightarrow \mathcal{C}(F) : H \mapsto \bigcup\{J \subseteq H \mid J \in \mathcal{C}(F)\}$ be the *control operator* for

$F \in \mathcal{F}_L$. The map κ_F is well defined by axiom (1). In the standard control technology, $\kappa_F(H) = sup\mathcal{C}_F(H)$, the supremal controllable sublanguage of H. We investigate how a control structure on L can be induced on M. The triple (L, θ, M) is *control consistent* if the following diagram commutes.

$$\begin{tikzcd}
\mathcal{F}_M \arrow[r, dashed, "\mathcal{C}_M"] & \mathcal{P}^2(M) \\
\mathcal{F}_L \arrow[u, "\theta"] \arrow[r, "\mathcal{C}_L"] & \mathcal{P}^2(L) \arrow[u, "\theta"']
\end{tikzcd}$$

The map $\mathcal{C}_M$ exists if and only if $\ker \theta \leq \ker(\theta \circ \mathcal{C}_L)$, which has the following paraphrase: if two low-level prefix-closed languages or 'subsystems' appear the same in the high level, then the sets of controllable sublanguages attached to these two languages must have the same image in the high level. We achieve this condition by appropriate design of T and θ. Since $\theta : \mathcal{F}_L \longrightarrow \mathcal{F}_M$ is surjective, the map $\mathcal{C}_M$ is unique if it exists. The term 'control consistency' is justified by the following proposition.

Proposition 1 *$\mathcal{C}_M$ is a control structure on M. In particular*

$$\begin{tikzcd}
\mathcal{P}(M) \arrow[r, "\kappa_M"] \arrow[d, "\theta^{-1}"'] & \mathcal{C}_M(M) \\
\mathcal{P}(L) \arrow[r, "\kappa_L"] & \mathcal{C}_L(L) \arrow[u, "\theta"']
\end{tikzcd}$$

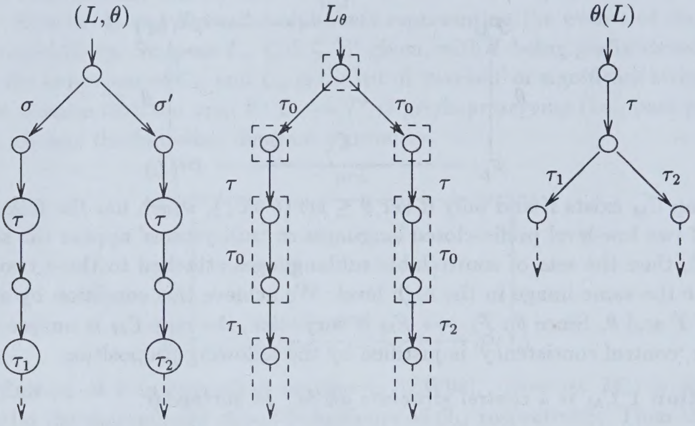
□

The commutative diagram in the above proposition captures the general notion of hierarchical consistency in [ZW90]. Given a high-level specification $E \subseteq M$, C_{hi} synthesizes $\kappa_M(E)$ in G_{hi}. Alternatively the high level specification E can be sent down via the 'command' channel as $\theta^{-1}(E)$. Then C_{lo} synthesizes $\kappa_L\theta^{-1}(E)$ in G_{lo}, and this synthesis is 'reported' via θ back to the high level. The commutative diagram ensures that the two paths of actions achieve the same objective in the high level. Thus hierarchical consistency ensures that the expectation in the high level, $\kappa_M(E)$, is consistent with the actual implementation via the low level, $\theta\kappa_L\theta^{-1}(E)$. This commutative diagram can be considered as an algebraic version of the diagram $con_{hi} = inf_{lohi} \circ con_{lo} \circ com_{hilo}$ in the block diagram describing the two-level hierarchy. The subscripts in $\mathcal{C}_M$ and $\mathcal{C}_L$ will be dropped if the context determines which is intended. Next we examine an 'observer' property of θ and its relation to the concept of observation equivalence, one of the main concepts in Milner's calculus of communicating systems [Mil80]. We represent L as a (deterministic) tree. Vocalize the nodes of the tree according to θ. Let $\tau_0 \notin T$ represent a silent transition and $T_0 := T \cup \{\tau_0\}$. From (L, θ), construct a tree, possibly nondeterministic, as follows:

1. retain all the nodes;
2. replace the label of a transition by τ_0 if the entrance node is silent;

3. replace the label of a transition by τ if the entrance node is vocal with the event τ.

Let L_θ be the tree so constructed from (L, θ). The following example illustrates the procedure.



Consider the tree L_θ as a transition structure with X_θ denoting the set of nodes of L_θ. Let $\delta_0 : T_0 \times X_\theta \longrightarrow \mathcal{P}(X_\theta)$ be the nondeterministic (partial) transition function of L_θ. This function can be extended to $\delta_0 : T_0^* \times X_\theta \longrightarrow \mathcal{P}(X_\theta)$ in a natural way. Let $P_0 : T_0^* \longrightarrow T^*$ be the natural projection which erases every τ_0 from a string. Then define $\delta : T^* \times X_\theta \longrightarrow \mathcal{P}(X_\theta)$ by $\delta(t,x) = \bigcup_{t_0}\{\delta_0(t_0,x) \,|\, P_0 t_0 = t\}$ where $t \in T^*$ and $x \in X_\theta$. Define $\delta_t : X_\theta \longrightarrow \mathcal{P}(X_\theta)$ with $\delta_t(x) = \delta(t,x)$ for $t \in T^*$. Let $\omega \in \mathcal{E}(X_\theta)$, the set of equivalence relations on X_θ, and $P_\omega : X_\theta \longrightarrow X_\theta/\omega$ be the corresponding canonical projection. Now define $\omega \circ \delta_t \in \mathcal{E}(X_\theta)$ as follows: for any $x_1, x_2 \in X_\theta$, $x_1 \equiv x_2 \, mod\,(\omega \circ \delta_t) \Longleftrightarrow P_\omega \delta_t(x_1) = P_\omega \delta_t(x_2)$. Following [Won76], we define the *observability congruence* $\omega_\theta \in \mathcal{E}(X_\theta)$ as

$$\omega_\theta := sup\,\{\omega \in \mathcal{E}(X_\theta) \,|\, \omega \leq ker\,\theta \wedge \bigwedge_{t \in T^*} \omega \circ \delta_t\}$$

Then θ is an *observer* if $\theta(L) \simeq L_\theta/\omega_\theta$ where $\simeq$ denotes isomorphism of transition structures. Intuitively the map θ is an observer if the silent transition in the low level cannot change the future options with respect to the high level. In another words the map θ is refined enough to distinguish critical branchings in the low level. In the example above, θ is not an observer because the future option is changed from $\{\tau, \tau\tau_1, \tau\tau_2, \cdots\}$ to $\{\tau, \tau\tau_1, \cdots\}$ by the silent transition σ. Hence ω_θ for this example, indicated by the dotted-line partition on L_θ, is strictly finer than ker θ. We have

Proposition 2

$$\theta^{-1} \circ pre_M = pre_L \circ \theta^{-1} \Longleftrightarrow \theta \text{ is an observer } \Longleftrightarrow \theta(L) \approx L_\theta$$

□

Here $\approx$ is the observation equivalence in [Mil80]. Now we present one of the main results in this paper: the preservation of the 'nonblocking' property between the two levels in the hierarchy. First we introduce an additional restriction on the control structure of L

$$(\forall K \in \mathcal{P}(H))\, \overline{K} \in \mathcal{C}(H) \Longrightarrow K \in \mathcal{C}(H)$$

where $H \in \mathcal{F}_L$. Again this condition holds in the standard control technology. Following [RW89], a language $E \subseteq L$ is *nonblocking* if $\overline{E} = \overline{\overline{E} \cap L_m}$. In words, a language E is nonblocking if any string in the prefix closure of E can always be extended to a marked string in the prefix closure of E: any task can always be continued to its completion. Then (L, θ) is *strongly observable* if $\theta|_K$, θ restricted to K, is an observer for any $K \in \mathcal{C}(L) \cap \mathcal{F}_L$. Intuitively (L, θ) is strongly observable if the observability is invariant under any action of control. Now we have

Proposition 3 *Suppose*

$$\theta^{-1}(M_m) = L_m$$
$$(L, \theta) \text{ is strongly observable}$$

Then

$$(\forall E \subseteq M)\, \kappa_M(E) \text{ is nonblocking} \iff \kappa_L \circ \theta^{-1}(E) \text{ is nonblocking}$$

□

The first condition ensures that the markings in the two levels are consistent. Next we introduce another closure operator, which will be convenient for expressing certain languages. If $H \subseteq L$, the *postfix closure of* H *in* L, denoted $pos_L(H)$, consists of all extensions of strings of H in L:

$$pos_L(H) = \{s \in L \,|\, (\exists u \in H)\, u \leq s\}$$

It is straightforward to check that, like the prefix closure operator, pos_L is a Kuratowski closure operator. Intuitively, the postfix closure of H in L embodies the 'evolutionary future' in L of words in H. A language H is *postfix closed in* L if $pos_L(H) = H$. As an operator, $pos_L \in Func(L)$. Again a natural question is whether

$$\theta \circ pos_L \overset{?}{=} pos_M \circ \theta$$

It turns out that we always have $\theta \circ pos_L \leq pos_M \circ \theta$. But the reverse inequality does not hold in general. The map θ is *postfix-preserving* (i.e., future-preserving) if $\theta \circ pos_L = pos_M \circ \theta$. The following proposition shows that 'θ is future-preserving' is logically equivalent to the property that θ is an observer.

Proposition 4

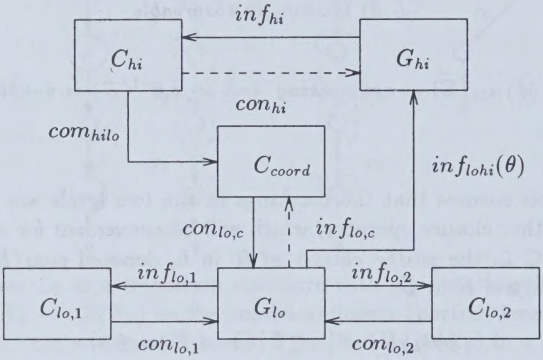
$$\theta \circ pos_L = pos_M \circ \theta \iff \theta^{-1} \circ pre_M = pre_L \circ \theta^{-1}$$

□

Note that there is an interesting duality between the past and future: the property that θ is an observer can be described either retrospectively or prospectively.

3 Modular Control and Coordination

In [RW88], the authors formalize a notion of modular control by exploiting the natural decomposition of control tasks. The main result in [RW88] states that if there is no conflict among the modular supervisors, then the behaviour under the concurrent supervisions of the modular supervisors is the same as that under a centralized supervisor. Here we consider the situation that the modular supervisors are conflicting and set up a coordinator which 'resolves' the conflict among the modular supervisors in a non-intrusive manner in a hierarchical structure. The architecture of the coordination can be displayed as follows:



Here $C_{lo,1}$ and $C_{lo,2}$ represent two low-level modular supervisors each synthesizing its own specification in G_{lo} independently, i.e., unaware of the existence of the other. Thus conflicts could result under the control of $C_{lo,1}$ and $C_{lo,2}$. For a remedy, the behaviour of G_{lo} under the concurrent supervisions of $C_{lo,1}$ and $C_{lo,2}$ is aggregated through the information channel as the behaviour of the high-level model G_{hi}. The high-level coordinator C_{hi} oversees the situation in the low level by examining the model G_{hi}. Once a critical situation is detected, in which any further execution of transitions thought of as 'safe' by both $C_{lo,1}$ and $C_{lo,2}$ will inevitably lead to conflicts, C_{hi} will 'disable' any such transition in the model G_{hi}. But recall that the control exercised by C_{hi} is only virtual. In actuality a command is sent down to a low-level coordination agent C_{coord} which will carry out the disablement so as to prevent G_{lo} from evolving into the conflict situation; the result of this disablement is reported back to C_{hi} by updating the model G_{hi}. Provided the aggregation is refined enough C_{hi} will be able to detect potential conflicts and rectify the situation in timely fashion.

Let $E_i \subseteq L$ be the specifications for $C_{lo,i}$ where $i = 1, 2$. Then $C_{lo,i}$ synthesizes $\kappa_L(E_i)$ in L. The behaviour of G_{lo} (in L) under the supervision of $C_{lo,i}$ is $\overline{\kappa_L(E_i)}$, and the behaviour under the concurrent supervisions of $C_{lo,1}$ and $C_{lo,2}$ is $\overline{\kappa_L(E_1)} \cap \overline{\kappa_L(E_2)}$. In general

$$\overline{\kappa_L(E_1 \cap E_2)} \subset \overline{\kappa_L(E_1)} \cap \overline{\kappa_L(E_2)}$$

with strict inclusion. The purpose of coordination is to restore the equality by reducing the right-hand side to the left. In addition we require that the coordinator be non-intrusive, i.e., that the coordinator exert control only when necessary. To be more precise, we make the following definition. Given $H \subseteq L$, let

$$\mathcal{B}(\overline{H}) := \{(s,\sigma) \in \overline{H} \times \Sigma \,|\, s\sigma \in L - \overline{H}\}$$

Then H_1, $H_2 \subseteq L$ are *non-redundant* if

$$\mathcal{B}(\overline{H_1}) \cap \mathcal{B}(\overline{H_2}) = \emptyset$$

Two non-redundant languages H_1 and H_2 do not share exiting events in L. If in addition H_1 and H_2 are controllable languages, the non-redundancy captures the intuitive notion that the control action of H_1 and H_2 are never redundant, i.e., never 'disable' the same event together. In a hierarchical structure (L, θ, M), a language $K \in \mathcal{C}(L)$ is *implementable via* θ *in* L if there is a language $N \in \mathcal{C}(M)$ such that $\overline{K} = \kappa_L \theta^{-1}(\overline{N})$. Intuitively a controllable language is implementable via θ if it can be realized by implementing the 'command' from the high level in a manner similar to that described in [ZW90] in the standard control technology, i.e., disabling the last possible controllable event. For $H \in \mathcal{F}_L$, a language $K \in \mathcal{C}(L)$ is *implementable via* θ *locally in* H if $\overline{K} \cap H$ is implementable via θ in H, i.e., $\overline{K} \cap H = \kappa_H \theta^{-1}(\overline{R})$ for some $R \in \mathcal{C}(\theta(H))$. Now we present the

Coordination problem: Given E_1, $E_2 \subseteq L$. Find $K \in \mathcal{C}(L)$ such that

$$\overline{\kappa_L(E_1 \cap E_2)} = \overline{\kappa_L(E_1)} \cap \overline{\kappa_L(E_2)} \cap \overline{K}$$

where K is non-redundant with respect to $\kappa_L(E_1)$ and $\kappa_L(E_2)$, and implementable via θ locally in a prefix-closed sublanguage of L.

◇

To make further progress we must introduce two additional restrictions on the control structure of L:

$$(\forall K \in \mathcal{C}(H))\, L - pos_L(H - \overline{K}) \in \mathcal{C}(L) \text{ where } H \in \mathcal{F}_L$$
$$\mathcal{C}(H) \cap \mathcal{F}_H \text{ is a complete sublattice of } \mathcal{P}(H) \text{ for every } H \in \mathcal{F}_L$$

The standard control technology again has these two properties. Now let $K_i := \kappa_L(E_i)$ for $i = 1, 2$ and

$$X := \theta(K_1) \cap \theta(K_2),\, Y := \overline{\theta(K_1)} \cap \overline{\theta(K_2)}$$

Assume

$$\theta^{-1}\theta(K_i) = K_i, \;\; i = 1, 2$$
$$(L, \theta) \text{ is strongly observable}$$

Then we have

Proposition 5 *The coordinator is given 'optimally' by*

$$\begin{aligned} \overline{K} &= \kappa_L \theta^{-1}(M - pos_M(Y - \overline{\kappa_Y(X)})) \\ &= L - pos_L[\theta^{-1}(Y) - \kappa_{\theta^{-1}(Y)} \theta^{-1}(\overline{\kappa_Y(X)})] \end{aligned}$$

Furthermore K *is implementable via* θ *locally in* $\theta^{-1}(Y)$.

□

Here 'optimality' means that for any coordinator K' we have $\overline{K'} \subseteq \overline{K}$. The languages $\theta(K_1)$ and $\theta(K_2)$ represent the objectives of the two modular supervisors as seen in the high level. The first condition ensures that these objectives are 'accurately reported'. The prefix closures of these two languages are their corresponding closed behaviours. Thus X and Y represent respectively the common objectives, and actual behaviours, of the modular supervisors. The first expression for $\overline{K}$ represents a global view of coordination. Intuitively $Y - \overline{\kappa_Y(X)}$ represents the set of strings in the model of the high-level coordinator that are conflicting or will lead to conflicting situations 'uncontrollably'. Then $pos_M(Y - \overline{\kappa_Y(X)})$ is the set of all possible futures of such conflicts, and $M - pos_M(Y - \overline{\kappa_Y(X)})$ is the 'largest' conflict-free behaviour. So the behaviour of the low-level coordinator, $\overline{K}$, can be thought of as the corresponding least restrictive 'implementation'. However, in actuality the coordination is done in a more local manner as described by the second expression for $\overline{K}$. Here the high-level coordinator, working locally in Y, is $\overline{\kappa_Y(X)}$. Its 'command' is implemented locally in $\theta^{-1}(Y)$ at the low level as $\kappa_{\theta^{-1}(Y)}\theta^{-1}(\overline{\kappa_Y(X)})$. It is the global behaviour of this implementation that is given by the second expression.

4 Conclusion

In this paper, we have formalized certain concepts pertaining to control architecture of DESs, especially that of hierarchical consistency with respect to nonblocking and coordination. The results are based on an abstract notion of control structure, which is independent of any specific control technology. It is found that the concept of observer [Won76] plays an essential role, as does that of postfix closure. Two directions of future studies present themselves: investigate these concepts when applied to specific control technologies, i.e., that of TDESs [BW92], and examine the possibility of weakening the observability concept to an appropriate notion of 'detectability'.

References

[BW92] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event systems. Technical Report 9210, Systems Control Group, Dept. of Elec. Eng., Univ. of Toronto, August 1992.

[LW88] F. Lin and W. M. Wonham. Decentralized supervisory control of discrete-event systems. *Information Sciences*, 44:199–224, 1988.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture notes on computer science.* Springer-Verlag, New York, 1980.

[RW88] P. J. Ramadge and W. M. Wonham. Modular supervisory control of discrete event systems. *Mathematics of Control, Signal and Systems*, 1(1):13–30, 1988.

[RW89] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. IEEE, Specical Issue on Discrete Event Dynamic Systems*, 77(1):81–98, January 1989.

[RW91] K. Rudie and Wonham. Think globally, act locally: decentralized supervisory control. Technical Report 9102, Systems Control Group, Dept. of Elec.

Eng., Univ. of Toronto, April 1991. IEEE Transaction on Automatic Control, November 1992, to appear.

[Won76] W. M. Wonham. Towards an abstract internal model principle. *IEEE Transaction on Systems, Man, and Cybernetics*, SMC-6(11):735–740, November 1976.

[Won91] W. M. Wonham. Some current research directions in control of discrete-event systems. *Preprint: European Control Conference*, July 1991.

[ZW90] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. Aut. Contr.*, 35(10):1125–1134, October 1990.

Modular Supervisory Control of Timed Discrete-Event Systems

B. A. Brandin W. M. Wonham B. Benhabib
brandin@control.utoronto.ca wonham@control.utoronto.ca beno@me.utoronto.ca
Computer Integrated Manufacturing Laboratory (CIMLab)
University of Toronto
Toronto, Canada M5S 1A4

Abstract

The timed discrete-event system framework [1, 2] admits disablement and forcing as means of control, as well as the treatment of hard temporal behavioural constraints. The notion of modular supervisory control of discrete-event systems proposed in [13] is extended to this framework. The approach consists of dividing the overall supervisory task into two or more subtasks, with the resulting individual subsupervisors being run concurrently to implement a solution of the original control problem.

A small manufacturing cell example illustrates the results.

1 Introduction

Discrete-event systems are dynamic systems that evolve in accordance with the abrupt occurrence of physical events. Such systems generally encompass processes that are discrete in time and state space, are often asynchronous, and are typically nondeterministic. To ensure the orderly occurrence of events in such classes of systems, i.e. according to given behavioural specifications, some degree of supervision and control is generally required.

Ramadge and Wonham initiated a systematic investigation[1] of discrete-event systems from a control theoretic point of view. Discrete-event systems are controlled as generators of a formal language; the adjunction of a control structure allows varying the language generated by the system within certain limits by accordingly enabling and disabling events; the desired performance of such controlled generators is specified by stating that their generated language must belong to some specification language.

Within this context, the design of supervisory control strategies relies on logic and formal language theory, and ensures that the plant under supervision will always behave *optimally* and *legally*, i.e. in the freest possible way within given behavioural specifications. Also, it is important to note that the supervisory control strategy thus obtained is correct by construction.

Timing introduces a new dimension of discrete-event system modelling and control, of considerable applied interest[2] but also of significant complexity. Many approaches are possible. The first that we know to have been proposed in the setting of [11] is that of Moller [6], who assigns time delays to events but does not explore their implication for control. Brave and Heyman [3] introduce time intervals for the possible occurrence of enabled events, and demonstrate how temporal features and logic based features of behaviour might be separated for independent treatment.

Brandin, Wonham and Benhabib [1, 2] incorporate in the framework of [11] the constraints that events may occur only within designated time bounds, relative to the times when the events become enabled. The model adjoins to the structure of [11] the timing features of TTM [7, 8, 9]. The resulting framework retains the concept of minimally restrictive supervision introduced in [11] and

(i) allows the treatment of hard temporal behavioural specifications, such as production time frames;

(ii) admits forcing and disablement as means of control;

(iii) allows the explicit incorporation in the system model of the effects of time related control enforcement constraints such as communication time delays between the supervisory control system and the system being supervised.

---

[1] For a review of work prior to 1989, please refer to [10].

[2] see below

In the framework of [1, 2], we discuss a modular approach to the synthesis of supervisors for timed discrete-event systems. The approach presented is based on [12, 13], and consists of dividing the overall supervisory task into two or more subtasks. Each of the latter is solved using the results pertaining to the supervisory control of timed discrete-event systems [1, 2], and the resulting individual subsupervisors are run concurrently to implement a solution of the original control problem. We refer to such a construction as a *modular synthesis*, and to the resultant supervisor as a *modular supervisor*.

Such constructions represent a very general approach to complex problems - sometimes called "divide and conquer". In addition to being more easily synthesized, a modular supervisor should ideally be more readily modified, updated and maintained. For example, if one subtask is changed, then it should only be necessary to redesign the corresponding component supervisor; in other words, the overall modular supervisor should exhibit greater flexibility than its 'monolithic' counterpart [12].

Unfortunately, these advantages are not always to be gained without a price. The fact that the individual supervisory modules are simpler implies that their control action must be based on an aggregated version of the global system state. A consequence of this relative insularity may be that different component supervisors, acting quasi-independently on the basis of partial information, come into conflict and the overall system fails to be nonblocking. Thus a fundamental issue that always arises in the presence of modularity is how to guarantee the nonblocking property of the final synthesis.

To this effect the state structure of a modular supervisor must be such that it embodies enough information to carry out its assigned control task, in a manner which does not conflict with control tasks concurrently implemented by other modular supervisors; in other words, the closed-loop behaviour resulting from the supervision of any modular supervisor must be in agreement with the closed-loop behaviour sought by any other modular supervisor working concurrently.

As an example in which blocking may occur due to the conflicting nature of control tasks consider the following scenario. Two numerically controlled machines (users) require the simultaneous use of a tool-bit and a fixture (resources); two modular supervisors are each responsible for the control of one resource. If these supervisors are such that at any one time each user is allowed to take possession of one resource (by the respective supervisor), blocking results since effectively either user prevents the other from finishing its task and freeing the resource it possesses.

2 Preliminaries

An *automaton* G is a 5-tuple $G = (\Sigma, Q, \delta, q_0, Q_m)$ in which Σ is an *alphabet*, $Q = \{q_0, q_1, ...\}$ is the *state set*, $\delta : \Sigma \times Q \to Q$ is the *transition function*, $q_0 \in Q$ is the initial state and $Q_m \subseteq Q$ is the *marked state set*. The *reachable state subset* of G is

$$Q_r = \{q \in Q | (\exists s \in \Sigma^*) \delta(s, q_0) = q\}; \tag{1}$$

G is *reachable* if $Q_r = Q$. The *coreachable state subset* is

$$Q_{cr} = \{q \in Q | (\exists s \in \Sigma^*) \delta(s, q) \in Q_m\}; \tag{2}$$

G is *coreachable* if $Q_{cr} = Q$. G is *trim* if it is both reachable and coreachable. The physically possible behavior or *closed behavior* of the plant is described by the following language

$$L(G) := \{s | s \in \Sigma^* \text{ and } \delta(s, q_0)!\} \tag{3}$$

where $\delta : \Sigma^* \times Q \to Q$ is the natural extension of $\delta : \Sigma \times Q \to Q$, and '!' means 'is defined'.

The behavior of the plant corresponding to the completion of a 'task', or *marked behavior*, is described by the language

$$L_m(G) := \{s | s \in \Sigma^* \text{ and } \delta(s, q_0) \in Q_m\}. \tag{4}$$

A discrete event system G is *nonblocking* if every reachable state is coreachable, i.e.

$$\overline{L_m(G)} = L(G). \tag{5}$$

Clearly, referring to the definition of a closed language, $L(G)$ is closed, but $L_m(G)$ need not be. It will be assumed that $L(G) \neq \emptyset$ and that $\overline{L_m(G)} = L(G)$.

Let $K \subseteq \Sigma^*$. We define the *closure* of K as $\overline{K} = \{t \in \Sigma^* | t \text{ is a prefix of } s \text{ for some } s \in K\}$. The language K is *L-closed* if $K = \overline{K} \cap L$.

Let K, L be arbitrary sublanguages of Σ^*. Then K, L are *nonconflicting* if

$$\overline{K \cap L} = \overline{K} \cap \overline{L} \tag{6}$$

Thus K and L are nonconflicting just in case every string that is both a prefix of K and a prefix of L can be extended to a word belonging to K and L in common.

Let $G = (\Sigma, Q, \delta, q_0, Q_m)$ be a timed discrete event system as described in [2]. The event set Σ is partitioned as follows

$$\Sigma := \Sigma_{rem} \dot\cup \Sigma_{spe} \dot\cup \{tick\} \tag{7}$$

where Σ_{rem} is the set of *remote* events, i.e. the set of events with an infinite upper time bound[3], Σ_{spe} is the set of *prospective* events, i.e. the set of events whose upper time bound is finite, and where the event *tick* represents the passing of one unit of clock time. Thus the temporal 'resolution' available for modelling purposes is just one unit of clock time.

As in [2] let $\Sigma_{hib} \subseteq \Sigma_{rem}$ be the set of *prohibitible* events, i.e. the set of events which through control action may be prevented indefinitely from occurring, and let $\Sigma_{for} \subseteq \Sigma_{rem} \dot\cup \Sigma_{spe}$ be the set of *forcible* events, i.e. the set of events able to preempt a *tick* of the clock. It is convenient to define the *uncontrollable* event set $\Sigma_{unc} := \Sigma_{spe} \dot\cup (\Sigma_{rem} - \Sigma_{hib})$, and the *controllable* event set $\Sigma_{con} := \Sigma_{hib} \dot\cup \{tick\}$. It should be noted that a forcible event may be controllable or uncontrollable.

Let $K \subseteq \Sigma^*$, $s \in K$ and write

$$Elig_K(s) = \{\sigma \in \Sigma | s\sigma \in \overline{K}\}. \tag{8}$$

A language $K \subseteq L(G)$ is defined [2] to be *controllable (with respect to G)* if, for all $s \in \overline{K}$,

$$Elig_K(s) \supseteq \begin{cases} Elig_G(s) \cap (\Sigma_{unc} \cup \{tick\}) & \text{if } Elig_K(s) \cap \Sigma_{for} = \emptyset \\ Elig_G(s) \cap \Sigma_{unc} & \text{if } Elig_K(s) \cap \Sigma_{for} \neq \emptyset. \end{cases} \tag{9}$$

Thus K controllable means that an event σ (in the full alphabet Σ including *tick*) may occur in K if either (i) it is uncontrollable or *tick*, and currently eligible in G, and no forcible event is currently eligible in K, or (ii) it is definitely uncontrollable, and eligible in G, but some forcible event is eligible in K. The effect of the definition is to allow the occurrence of *tick* (when it is eligible in G) to be ruled out of K only when a forcible event is eligible in K and could thus be relied on to preempt it.

Let $K_1, K_2 \subseteq L(G)$. K_1 and K_2 are said to be *jointly coercive* if $\forall s \in \overline{K_1 \cap K_2}$

$$tick \in Elig_{L(G)}(s) - Elig_{K_1 \cap K_2}(s) \Rightarrow Elig_{K_1 \cap K_2}(s) \cap \Sigma_{for} \neq \emptyset, \tag{10}$$

Thus K_1 and K_2 are jointly coercive if, in case the occurrence of *tick* must be ruled out of K_1 or K_2 or both, there exists a forcible event eligible in both K_1 and K_2 that can be relied on to preempt it.

While introducing time in the modular supervisory control context, we assume that the various modular supervisors have access to a global clock[4], and thus share information on the passing of clock time. As mentioned above, in the framework of [1, 2] the passing of one unit of clock time is represented by the discrete event *tick*, no different in nature from any other discrete event. Intuitively[5] then, the introduction of time should not overly complicate the modular synthesis problem, but possibly enlarge[6] the class of legal closed-loop behaviours: the inclusion of timing information may only augment the information available for control purposes by allowing controls to be generated upon the occurrence of strings $s \in (\Sigma_{rem} \dot\cup \Sigma_{spe} \dot\cup \{tick\})^*$ as opposed to the occurrence of strings $s \in (\Sigma_{rem} \dot\cup \Sigma_{spe})^*$ only.

3 The Conjunction of Supervisors

This section treats the conjunction of supervisors. Let S_1 and S_2 be proper supervisors for G: that is, each of S_1 and S_2 is a trim automaton, is complete for G (equivalently, $L_m(S_1)$, $L_m(S_2)$ are controllable with respect to G), and each of S_1/G, S_2/G is nonblocking, namely

$$\overline{L_m(S_1/G)} = L(S_1/G), \qquad \overline{L_m(S_2/G)} = L(S_2/G)$$

---

[3] Recall that in [2] every event $\sigma \in \Sigma_{rem} \dot\cup \Sigma_{spe}$ is equipped with a *lower time bound* l_σ and an *upper time bound* u_σ; the triples $(\sigma, l_\sigma, u_\sigma)$ are referred to as *timed events*.

[4] This assumption is easily satisfied in any computer based implementation (e.g.[1]) in which a number of computers host corresponding supervisory control modules.

[5] also proven below

[6] but never reduce

Recalling the definitions of reachable subautomaton and of product automaton, we define the *conjunction* of S_1 and S_2, written $S_1 \wedge S_2$, as the reachable subautomaton of the product $S_1 \times S_2$. It is easily seen from the definition that the supervisory action of $S_1 \wedge S_2$ is to enable an event σ just when σ is enabled by S_1 and S_2 simultaneously. To describe the action of $S_1 \wedge S_2$ more fully we have the following.

Theorem 1

Under the foregoing conditions $L_m((S_1 \wedge S_2)/G) = L_m(S_1/G) \cap L_m(S_2/G)$. Furthermore, $S_1 \wedge S_2$ is a proper supervisor for G if and only if it is trim and $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting.

Proof

We have

$$\begin{aligned} L_m((S_1 \wedge S_2)/G) &= L_m(S_1 \wedge S_2) \cap L_m(G) \\ &= L_m(S_1 \times S_2) \cap L_m(G) \\ &= L_m(S_1) \cap L_m(S_2) \cap L_m(G) \\ &= L_m(S_1/G) \cap L_m(S_2/G). \end{aligned} \tag{11}$$

Let us now consider the second statement. Since $L(S_1 \wedge S_2) = L(S_1) \cap L(S_2)$, we have $L((S_1 \wedge S_2)/G) = L(S_1/G) \cap L(S_2/G)$. So $S_1 \wedge S_2$ is nonblocking if and only if $L_m(S_1/G)$ and $L_m(S_2/G)$ are nonconflicting.

Now S_1 and S_2 proper implies that each is complete by definition. So $L(S_1)$ and $L(S_2)$ are controllable with respect to G. Since $L(S_1) \cap L(S_2)$ is controllable, $S_1 \wedge S_2$ is complete. Thus $S_1 \wedge S_2$ is a proper supervisor if and only if trim by definition. □

Theorem 2

Let $K_1, K_2 \subseteq L(G)$ be controllable with respect to G. If K_1 and K_2 are nonconflicting, and jointly coercive, then $K_1 \cap K_2$ is controllable with respect to G.

Proof

Let K_1 and K_2 be nonconflicting, i.e. $\overline{K_1} \cap \overline{K_2} = \overline{K_1 \cap K_2}$. Let $s \in \overline{K_1} \cap \overline{K_2}$. Finally, let $K_1, K_2 \subseteq L(G) \subseteq \Sigma^*$ be controllable with respect to G. Then for $i = 1, 2$ we have $(\forall s \in \overline{K_i})$

$$Elig_{K_i}(s) \supseteq \begin{cases} Elig_{L(G)}(s) \cap (\Sigma_{unc} \cup \{tick\}) & \text{if } Elig_{K_i}(s) \cap \Sigma_{for} = \emptyset \\ Elig_{L(G)}(s) \cap \Sigma_{unc} & \text{if } Elig_{K_i}(s) \cap \Sigma_{for} \neq \emptyset, \end{cases} \tag{12}$$

or equivalently

$$Elig_{K_i}(s) \supseteq Elig_{L(G)}(s) \cap \Sigma_{unc} \wedge (tick \in Elig_{K_i}(s) \vee Elig_{K_i}(s) \cap \Sigma_{for} \neq \emptyset). \tag{13}$$

We note that $\overline{K_1} \cap \overline{K_2} = \overline{K_1 \cap K_2}$ implies $Elig_{K_1}(s) \cap Elig_{K_2}(s) = Elig_{K_1 \cap K_2}(s)$. Then

$$Elig_{K_1}(s) \supseteq Elig_{L(G)}(s) \cap \Sigma_{unc} \text{ and } Elig_{K_2}(s) \supseteq Elig_{L(G)}(s) \cap \Sigma_{unc}$$

imply

$$Elig_{K_1 \cap K_2}(s) \supseteq Elig_{L(G)}(s) \cap \Sigma_{unc}.$$

Finally, consider the following two cases

(i) Let $tick \in Elig_{K_1}(s)$ and $tick \in Elig_{K_2}(s)$. Then $tick \in Elig_{K_1 \cap K_2}(s)$.

(ii) Let $tick \in Elig_{L(G)}(s) - Elig_{K_1 \cap K_2}(s)$, i.e. the occurrence of *tick* must be ruled out of K_1 or K_2 or both. Then, there exists a forcible event eligible in both K_1 and K_2 that can be relied on to preempt it, i.e. $Elig_{K_1 \cap K_2}(s) \cap \Sigma_{for} \neq \emptyset$.

Consequently,

$$Elig_{K_1 \cap K_2}(s) \supseteq Elig_{L(G)}(s) \cap \Sigma_{unc} \wedge (tick \in Elig_{K_1 \cap K_2}(s) \vee Elig_{K_1 \cap K_2}(s) \cap \Sigma_{for} \neq \emptyset), \tag{14}$$

or equivalently

$$Elig_{K_1 \cap K_2}(s) \supseteq \begin{cases} Elig_{L(G)}(s) \cap (\Sigma_{unc} \cup \{tick\}) & \text{if } Elig_{K_1 \cap K_2}(s) \cap \Sigma_{for} = \emptyset \\ Elig_{L(G)}(s) \cap \Sigma_{unc} & \text{if } Elig_{K_1 \cap K_2}(s) \cap \Sigma_{for} \neq \emptyset, \end{cases} \tag{15}$$

i.e. $K_1 \cap K_2$ is controllable with respect to G. □

Theorem 3

Let $E_1, E_2 \subseteq L(G)$. If $supC(E_1)$ and $supC(E_2)$ are nonconflicting, and jointly coercive, then $supC(E_1 \cap E_2) = supC(E_1) \cap supC(E_2)$.

Proof

The proof is similar to the proof of Theorem 6.1 of [13]. Note that $supC(E_1) \subseteq E_1$ and $supC(E_2) \subseteq E_2$ together imply that $supC(E_1) \cap supC(E_2) \subseteq E_1 \cap E_2$. From Theorem 2 above, since $supC(E_1)$ and $supC(E_2)$ are each controllable, nonconflicting, and jointly coercive, $supC(E_1) \cap supC(E_2)$ is controllable, and thus $supC(E_1 \cap E_2) \supseteq supC(E_1) \cap supC(E_2)$. For the reverse inclusion, we use the fact that for $L_1, L_2 \subseteq L(G)$, if $L_1 \subseteq L_2$, then $supC(L_1) \subseteq supC(L_2)$. Since $E_1 \cap E_2 \subseteq E_1$, we have that $supC(E_1 \cap E_2) \subseteq supC(E_1)$. Similarly $supC(E_1 \cap E_2) \subseteq supC(E_2)$. Thus $supC(E_1 \cap E_2) \subseteq supC(E_1) \cap supC(E_2)$. □

4 Example

Let us consider the small manufacturing cell shown in Figure 1. Two machines M_1 and M_2 and a buffer of size one constitute the cell. The machines are represented in the form of activity transition diagrams in Figure 2.

We have $\Sigma_{unc} = \{\beta_1, \beta_2, \lambda_1, \lambda_2\}$, $\Sigma_{hib} = \{\alpha_1, \alpha_2, \mu_1, \mu_2\}$ and $\Sigma_{for} = \{\alpha_1, \alpha_2\}$. The following timing information applies

$$(\alpha_1, 1, \infty) \quad (\beta_1, 1, 2) \quad (\lambda_1, 0, 2) \quad (\mu_1, 1, \infty) \quad (\alpha_2, 1, \infty) \quad (\beta_2, 1, 1) \quad (\lambda_2, 0, 1) \quad (\mu_2, 2, \infty).$$

The corresponding timed transition graphs are obtained as illustrated in [1, 2], and are shown[7] in Figure 3. The workpieces produced by M_1 are placed in the buffer and are consequently available for further work by M_2. Both machines may either be idle, working or down. Once a work cycle has begun, the machines either finish working or breakdown, in which case they are repaired. The following production specifications are considered:

(i) the buffer must not overflow or underflow,

(ii) if both machines are broken down, then M_2 must be repaired before M_1.

Centralised supervisory control [1, 2] is considered first. The cell open loop behaviour G (49 states, 120 transitions), is obtained by composing[8] M_1 and M_2.

The above mentioned specifications are translated in the form of automata, i.e. we introduce the automata R_1 and R_2 which enforce the buffer and breakdown specifications respectively. R_1 and R_2 are shown in Figure 4. Accordingly, the corresponding specification languages are provided by $E_1 = L_m(R_1)$ and $E_2 = L_m(R_2)$ respectively. R_1 and R_2 are combined together through the *meet* operation[9] $\wedge$, i.e. $R_3 = R_1 \wedge R_2$. R_3 represents the combined specifications and is shown in Figure 5. Accordingly we have $E_3 = E_1 \cap E_2 = L_m(R_3)$. The cell closed-loop behaviour which meets the specifications in the freest possible way is given by $supC(E_3)$, i.e. the supremal controllable sublanguage of the cell with respect to the combined specifications. The corresponding automaton S, such that $L(S) = supC(E_3)$, has 52 states and 106 transitions.

One possible modular design solution is now proposed. The supervisory task is carried out concurrently by two modules: the first module is responsible for the buffer specification; and the second module is responsible for the breakdown specification.

Let G represent the parallel composition of the two machines M_1 and M_2. We reintroduce the automata R_1 and R_2, shown in Figure 4, which enforce the buffer specification and breakdown specification respectively.

It can be checked that $E_1 = L_m(R_1)$ and $E_2 = L_m(R_2)$ are controllable with respect to G. Hence R_1 and R_2 are complete. Furthermore, E_1 and E_2 can be shown to be nonconflicting with respect to $L(G)$, and thus that R_1 and R_2 are nonblocking for G; so we may conclude from Theorem 1, that R_1 and R_2 are proper supervisors for G.

---

[7] In figures, the *tick* event will be labeled t.

[8] cf. [1]

[9] Here and below, $G_3 = G_1 \wedge G_2$ denotes the operation that returns a reachable TDES G_3 such that $L_m(G_3) = L_m(G_1) \cap L_m(G_2)$, $L(G_3) = L(G_1) \cap L(G_2)$.

For our modular supervisor, we take the conjunction

$$R_3 = R_1 \wedge R_2.$$

It is easily checked that $E_1 = L_m(R_1)$ and $E_2 = L_m(R_2)$ are nonconflicting, so that R_3 is trim; furthermore it can be checked that $E_3 = L_m(R_3) = L_m(R_1) \cap L_m(R_2)$ is both controllable and nonconflicting with respect to $L(G)$, and thus it can be deduced that R_3, shown in Figure 5, is a proper supervisor for G.

Finally, we consider the case in which the specification language is not controllable with respect to G. We introduce the automaton R_1' to enforce the buffer specification. R_1' is shown in Figure 6. Note that $E_1' = L_m(R_1')$ is not controllable with respect to G. It can be checked that $sup\mathcal{C}(E_1' \cap L(G))$ and $sup\mathcal{C}(E_2 \cap L(G))$ satisfy the assumptions of Theorem 3, and thus we have that

$$sup\mathcal{C}(E_1' \cap L(G)) \cap sup\mathcal{C}(E_2 \cap L(G)) = sup\mathcal{C}(E_1' \cap E_2 \cap L(G)).$$

The corresponding automata S_1', such that $L_m(S_1') = sup\mathcal{C}(E_1' \cap L(G))$, and S_2, such that $L_m(S_2) = sup\mathcal{C}(E_2 \cap L(G))$, respectively have 58 states and 123 transitions, and 49 states and 117 transitions.

5 Conclusions

The results on modular supervisory control of discrete-event systems presented by Ramadge and Wonham [13] are shown to require minor modifications in the timed discrete-event system context introduced by Brandin, Wonham and Benhabib [1, 2]. The framework obtained provides us with all the advantages related to modular supervision, but also

(i) allows the treatment of hard temporal behavioural specifications;

(ii) admits forcing and disablement as means of control;

(iii) allows the explicit incorporation in the system model of the effects of time related control enforcement constraints such as communication time delays between the supervisory control system and the system being supervised.

A small manufacturing cell example illustrates the results presented.

References

[1] B. A. Brandin, W.M. Wonham and B.Benhabib, "Manufacturing cell supervisory control - a timed discrete-event system approach", 1992 IEEE International Conference on Robotics and Automation, Nice, May 1992, pp. 531-536.

[2] B. A. Brandin and W.M. Wonham, "The Supervisory Control of Timed Discrete-Event Systems", Systems Control Group Report no. 9210, Department of Electrical Engineering, University of Toronto, 1992.

[3] Y. Brave and M. Heymann, "Formulation and control of real time discrete event processes", Proc. 27th Conference on Decision and Control, Austin, Texas, December 1988, pp. 1131-1132.

[4] S. B. Gershwin, "A hierarchical framework for discrete event scheduling in manufacturing systems", Proc. IIASA Workshop on Discrete Event Systems, Sopron, Hungary, August 1987. In P. Varaiya, A. B. Kurzhanski Eds., Discrete Event Systems: Models and Applications. Lecture Notes in Control and Information Sciences 103, Springer Verlag, Berlin, 1988, pp. 197-216.

[5] C. H. Golaszewski and P. J. Ramadge, "Control of discrete event processes with forced events", Proc. 26th Conference on Decision and Control, Dec. 1987, pp. 247-251.

[6] P. Moller, "Introducing real time in the algebraic theory of finite automata", Working Paper WP-86-49, International Institute for Applied System Analysis, Laxenburg, Austria, September 1986.

[7] J. S. Ostroff, "Deciding properties of timed transition models", IEEE, Trans. on Parallel and Distributed Systems, Vol. 1, No. 2, April 1990, pp. 170-183.

[8] J. S. Ostroff and W. M. Wonham, "A framework for real-time discrete event control", IEEE, Transactions on Automatic Control, Vol. 35, No. 4, April 1990, pp. 386-397.

[9] J. S. Ostroff, "Temporal Logic for Real-Time Systems", Research Studies Press Ltd, 1989

[10] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems", IEEE, Proc., Vol. 77, No. 1, January 1989, pp. 81-98.

[11] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event systems", SIAM, J. Control and Optimization, Vol. 25, No. 1, Jan. 1987, No. 1, pp. 206-230.

[12] W.M. Wonham, "Notes on Control of Discrete-Event Systems", Department of Electrical Engineering, University of Toronto, Revised 910826.

[13] W.M. Wonham and P.J. Ramadge, "Modular supervisory control of discrete event systems", Maths. of Control, Signals and Systems, Vol. 1, No. 1, 1988, pp. 13-30.

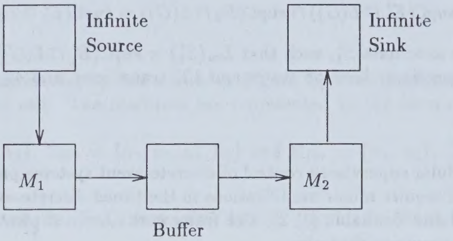
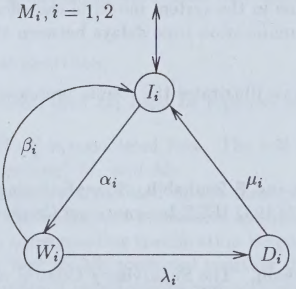


Fig. 1 Two-machine workcell setup.



I_i : *M_i is idle* , W_i : *M_i is operating,* D_i : *M_i is broken down*

α_i : *initialise machine M_i operation,* β_i : *M_i operation finish* ,
λ_i : *machine M_i failure* , μ_i : *M_i repair,*

Fig. 2 The activity transition graphs for $M_i, i = 1, 2$

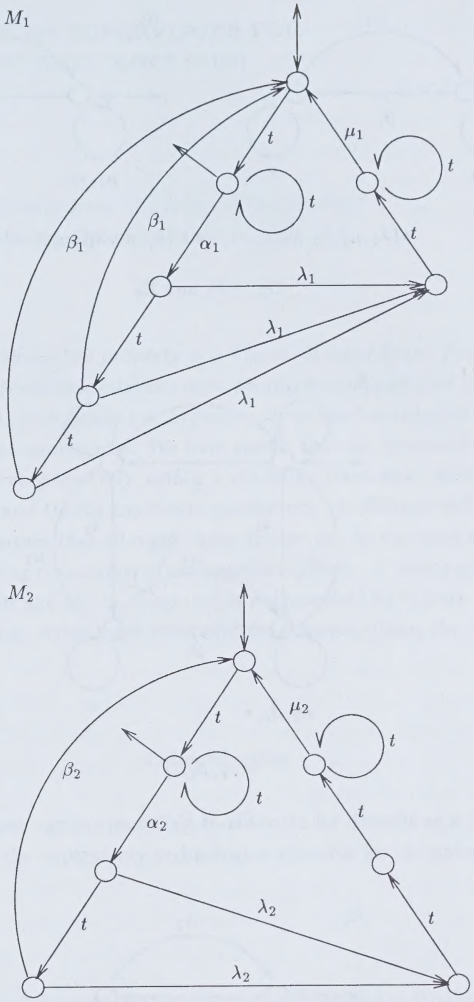
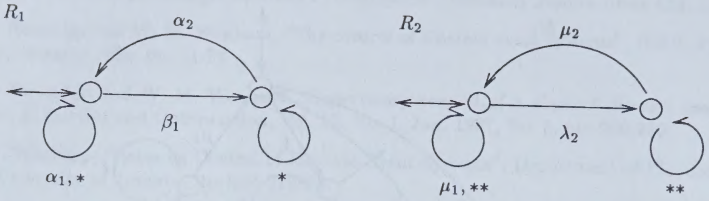
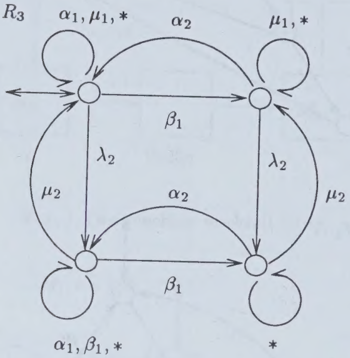

M_1
t
μ_1
t
t
β_1
β_1
α_1
t
λ_1
t
λ_1
λ_1
t
M_2
t
μ_2
t
β_2
t
α_2
t
λ_2
t
t
λ_2

Fig. 3 The timed transition graphs of M_1 and M_2


R_1
α_2
β_1
$\alpha_1, *$
$*$
R_2
μ_2
λ_2
$\mu_1, **$
$**$

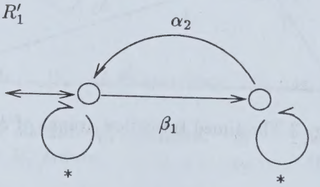
$*\{\lambda_1, \mu_1, \beta_2, \lambda_2, \mu_2, t\}$ $**\{\alpha_1, \beta_1, \lambda_1, \alpha_2, \beta_2, t\}$

Fig. 4 R_1 and R_2


R_3
$\alpha_1, \mu_1, *$
α_2
$\mu_1, *$
β_1
λ_2
λ_2
μ_2
α_2
μ_2
β_1
$\alpha_1, \beta_1, *$
$*$

$*\{\lambda_1, \beta_2, t\}$

Fig. 5 R_3


R_1'
α_2
β_1
$*$
$*$

$*\{\alpha_1, \lambda_1, \mu_1, \beta_2, \lambda_2, \mu_2, t\}$

Fig. 6 R_1'

ON THE SYNTHESIS OF SUPERVISORS FOR TIMED DISCRETE EVENT PROCESSES†

S.D. O'Young

Systems Control Group, Department of Electrical Engineering
University of Toronto, Toronto, Ontario, Canada M5S 1A4

ABSTRACT *The dynamical property of a Timed Discrete Event Process (TDEP) is described as a set of state transition relations with special* `tick` *transitions to track the passage of time. A real-time supervisor delays and expedites the actuation requests from the plant controller in order to exercise supervision. We have shown that the proposed delay and expedite technologies could work cooperatively within a modeling framework which is capable of expressing safety, liveness and timing constraint persistency. Additional tick-coreachability and escapability conditions ensure that all supervisory actions can be executed in a finite number of steps and within the timing constraints of the specified system. A least-restrictive nonblocking real-time supervisor exists and can be computed in polynomial* $O(n^3)$ *time, where* n *is the state size of the specified system. With some mild additional assumptions, the time-complexity can be reduced to* $O(n)$*.*

1. Introduction

The objective is to explore various modeling frameworks for describing a timed discrete event process and to examine the supervisory technologies available for the enhancement of its real-time performance.

1.1 Supervisory Tasks

We begin by examing the information flow structure of a Ramadge-Wonham [1] discrete-event supervisory control system where all transitions are reported by the plant to the controller and the supervisor via a common signal communication channel. An RW supervisor (fig. 1.1.1) arbitrates the controller's actions by refusing, if necessary, its actuation requests. Uncontrollable events are autonomous transitions taking place in the plant without having been actuated by the supervisor. A conventional feedback control system (fig. 1.1.2) can be regarded as an unsupervised RW system where all actuation requests are automatically granted.

---

† The research has been supported by the Information Technology Research Centre of Ontario, Canada, Bell Northern Research Ltd., Canada and Condata Technologies Ltd., Canada.

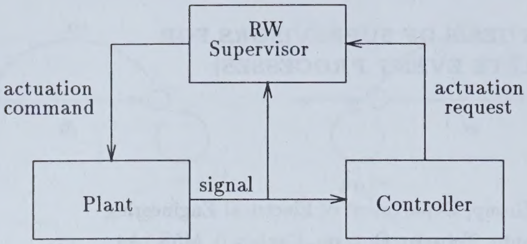

RW Supervisor
actuation command
actuation request
Plant
signal
Controller

fig. 1.1.1

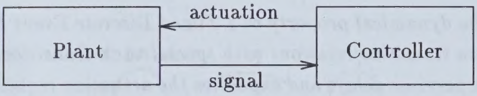

actuation
Plant
Controller
signal

fig. 1.1.2

In addition to the passive filtering of the actuation requests, our real-time supervisor (fig. 1.1.3) takes on the active role of sending *timely* actuation warnings back to the plant controller for expediting certain control actions. Hence, the supervisor can ensure that specified liveness requirements be met: a time-limited task must be performed within the specified time. The arbitration task of our real-time supervisor has also been made time-sensitive: the supervisor might grant a previously disabled actuation request at a later time without having been updated, in the interim, with new information from the plant. An RW supervisor would only revise its disablement pattern upon a change of state (an event) from the plant, otherwise its disablement pattern stays constant w.r.t. time.

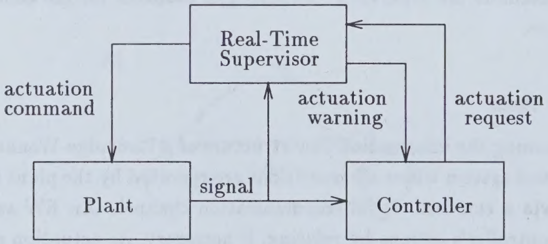

Real-Time Supervisor
actuation command
actuation warning
actuation request
Plant
signal
Controller

fig. 1.1.3

1.2 Modeling of Time

Since time is involved in the supervisory action, our modeling framework must involve mechanisms for detecting the passage of time. Let's first examine two basic systems for modeling time within a state transition structure. Consider a fragment of a state transition structure as shown in fig. 1.2.1. modeling some discrete event process where it has just entered state 1 and the transition from state 1 to 2 is labeled by the event α. Let l and u be its integer-valued lower and upper time bounds, respectively. Here, we have the choices of using a point time system or an interval time system. In the point time system, the passage of time is measured by counting the ticking of a global clock, and a lower time bound of l dictates that the process must wait in state 1 until l ticks have been counted before α can occur. For the interval time system, local clocks are used instead of a single global clock. The process must simply stay in state 1 for l units of time before α can occur. This is accomplished via the use of a local clock which starts running when the process enters state 1. The timing difference between the two systems is most acute when $l = 1$: for the point time system, α can occur almost immediately if the global clock happens to tick right after the process enters into state 1. For the interval time system, the process must always wait one unit of time before α is eligible to occur. For simplicity, we have chosen the point time system but our results can be extended to the interval time system with some additional modeling overhead. The reader is referred to Wong-Toi and Hoffmann [2] for an exposition on supervisory control using the interval time system and to a tutorial on the formal methods for the specification of time for control design by Ostroff [3].

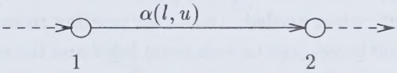


fig. 1.2.1

1.3 Window of Opportunity

The window of opportunity for the occurrence of a timed state transition, as demarcated by the lower and upper time bounds, can be considered as either passive or active. For a passive window, if the transition ever occurs, then it must take place within the timing window. For an active window, the transition must occur within the window unless its eligibility has been preempted by another transition. Consider the active-window timed process in fig. 1.3.1. Event α must occur after the lth and before the $(u+1)$th clock tick unless β occurs before α, taking the process to state 3 where α cannot occur. Brave and Heymann [4] have chosen the passive window semantics for their real-time counter automaton and Ostroff [5,6] uses the active window semantics in his Timed Transition Models (TTM). We have adopted the active window semantics for its ability to express the liveness requirement that something (good) must happen within a specified time. Moreover, the passive window semantics can be modeled by the active one as shown in fig. 1.3.2 where the γ transition is forced to occur between the uth and $(u+1)$th clock tick, taking the process to state 3 where the α transition is no longer eligible to occur.

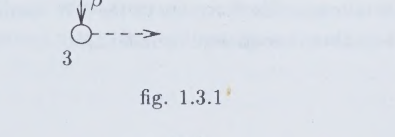


fig. 1.3.1

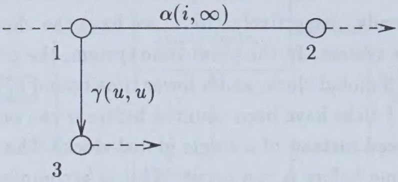


fig. 1.3.2

1.4 Persistency of Timing Constraints

The choice is on whether or not to re-initialize all timing constraints upon the occurrence of *each* state transition. Our modeling frame allows the timing constraints to persist over state transitions as shown in fig. 1.4.1 where the timing constraints on β is unaffected by the occurrence of α. Such a persistency rule allows us to express an on-going timing constraint such as the fuel burn-up time of a rocket over a sequence of internal state changes throughout its flight. Brave and Heymann [4] reset the tick counter in their counter automaton after each state transition, and Wong-Toi and Hoffmann [2] allows all timers to be active at all time and a particular timer is reset only when needed. Our timer reseting rules follow the Ostroff's TTM [5,6] semantics where a timer is assigned to each event label and the enablement of the counting coincides with the enablement of the corresponding event. Persistency over state transitions can be modeled within this framework. It should be noted the validity our synthesis results is independent of the choice of a particular persistency rule.

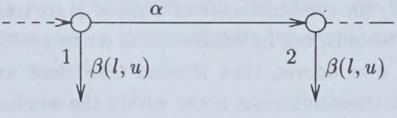


fig. 1.4.1

1.5 Supervisory mechanisms

Our supervisory technology is prescribed according to the direction of information between the supervisor and the plant controller as shown in fig. 1.1.3. The incoming actuation request received from the controller is managed by a delay mechanism and outbound warning sent to the controller is implementated as an expediting mechanism.

The supervisor delays a certain *delayable* transition by not granting its actuation so as to enforce certain safety requirements. The maximum delayable time is dictated by the upper time bound prescribed on the transition. When the upper time bound is taken to be infinity, then the delay can be made indefinite; this special case corresponds to the RW disablement mechanism. Our delay technology is, therefore, a timed version of the RW disablement mechanism where disablement is allowed to be either time-limited or indefinite.

The idea behind the expediting supervisory mechanism is to "beat the clock" by ensuring something (good) occur before the pending tick, thus rendering the tick transition exiting the present state of the process controllable in the RW sense. Consider the transition structure in fig. 1.5.1 where both α and β are both *expeditible*, i.e. they can be forced to occur before the next clock tick. The supervisory can preempt the tick transition from occurring, while the supervised process is in its present state, by sending a warning to the controller requiring it to issue an actuation request for either α or β before the next tick. This expediting mechanism can be regarded as a timed version of the Golaszewski and Ramadge forcing mechanism [7] in the context of their nondeterministic supervisor.

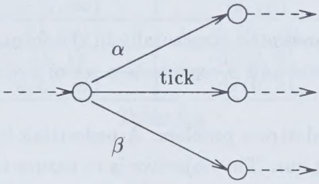


fig. 1.5.1

1.6 Supervisory Synthesis

Real-time supervisory synthesis for discrete event systems for guaranteeing safety have been studied by Golaskewski and Ramadge [8], Ostroff [9] and more recently by Wong-Toi and Hoffmann [2] where the sole technology for supervision is disablement. Forcing as an additional control technology was first proposed by Golaskewski and Ramadge [7] in a non-time setting, and was incorporated into a real-time supervision by Brave and Haymann [4] using their counter automata.

Our main contribution is to show that both forcing and disablement strategy can work cooperatively within a general discrete event process modeling framework where safety, liveness and timing constraint persistency properties can be expressed. The controllable subprocesses are shown to be close under arbitrary union such that they form a semi-lattice where an supremal element exists. We give a polynominal time algorithm for finding the supermal element.

Our supervisory synthesis theory is state-based as opposed to the more conventional formal language based [1,10]. The main reason for adopting a state-based model is to allow the explicit reference to program variables, such as the values of the tick counters. We have linked the supervisory (delay and expediting) technologies to the assignment of tick counters; hence their characteristics can be generalized to be time-varying. For example, a delayable event is controllable in the RW sense only up to a certain maximum allowable time, after which it becomes uncontrollable. A secondary motivation is the direct connection between state models and computational data structures since all computer-aided supervisory synthesis routines [11,12] are programmed using (finite) state representations, e.g. finite state generators of formal languages.

Brandin *et als.* [13,14] have developed a formal language version our synthesis theory using their Timed Ramadge Wonham (TRW) models. They have retained the RW control technology where the disablement mechanism is restricted to time-invariant: a controllable event *must* have an infinite upper time bound. This is necessary because there is no direct reference to program variables within their formal language setting. A condition of no "tick-free-loop" is also imposed on their TRW models as a sufficient condition for avoiding infinite number of state transitions between clock ticks. This restriction is avoided in our state-based approach.

2. A Supervisory Control Example

A specified system can be represented structurally in the form of a synchronous product [15] $P := G \|_S S$ where G is the plant and S represents a set of dynamical constraints on G.

Consider the following bus-pedestrian problem. A pedestrian is crossing a road and his life is threatened by an approaching bus. The objective is to ensure that the pedestrian survives by the timely execution of the supervisory commands for forcing him to jump off the road and/or stopping the bus.

The activity Discrete Event Processes (DEPs) of the bus (G_a^{bus}) and the pedestrian (G_a^{ped}) are shown in fig. 2.1 and 2.2. Suppose the lower and upper time bounds for the event **pass** are l_{pass} and u_{pass}, respectively; i.e. the bus would not pass until l_{pass} clock ticks have lapsed, and must pass before $(u_{pass} + 1)$th clock tick occurs. The lower and upper time bounds for the event **jump** are l_{jump} and u_{jump} respectively; i.e. the pedestrian must stay on the road for at least l_{jump} clock ticks for the purpose of crossing it, and after l_{jump} clock ticks, he is close enough to the curb on the opposite side that he can jump off the road if necessary.

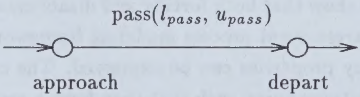


Fig. 2.1

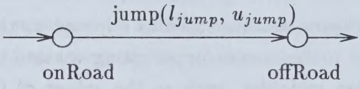


Fig. 2.2

Since there are no shared non-tick event labels, and hence no shared dynamics between the bus and pedestrian, their composite dynamics can be expressed as the synchronous product of the respective DEPs. That is, the plant $G := \mathbf{Tdep}\,(G_a^{bus}, T_a^{bus}) \|_S \mathbf{Tdep}\,(G_a^{ped}, T_a^{ped})$, where T_a^{bus} and T_a^{ped} are the timing constraints on the bus and on the pedestrian, respectively and function **Tdep** represents the forming a timed state transition structure from a logical activity process and a set of timing constraints. To ensure the survival of the pedestrian, we must avoid the states where the pedestrian is (or more appropriately was) on the road (`onRoad`) and the bus has run (`depart`) over him, and such states form the bad state set X_b of G. We will compute the largest controllable and State Avoiding Nonblocking Escapable (SANE) Discrete Event SubProcess (DESP) [16] of G for various supervisory technologies and timing constraints as tabulated in table 5.3 where the symbols d and e designate the transition to be delayable and expeditible, respectively.

case	jump	pass	figure (app)
1	e, $(1, \infty)$	$(2, \infty)$	A
2	$(1, \infty)$	d, $(0, \infty)$	B
3	$(1, 1)$	d, $(0, 2)$	C
4	e, $(1, \infty)$	d, $(0, 2)$	D

Table 5.3

Case 1: Suppose that the brakes of the bus have malfunctioned, and the only supervisory technology is expediting; i.e. the pedestrian can be told (or forced) to jump before the bus passes. In other words, the event `jump` is considered expeditible. The pedestrian can be ensured to survive if he is told to jump after the first clock tick and before the second tick. If the lower time bound of the event `pass` is reduced to 1, then the largest controllable and SANE DESP of P is empty, and the safety of the pedestrian cannot be guaranteed.

Case 2: Suppose that pedestrian is wearing a headset playing overwhelmingly loud music that he is not able to hear the warnings from the supervisory system, If the brakes of the bus are functioning, we can change the supervisory technology to delay: the bus driver is willing to stop and let the pedestrian cross safely by making the event `pass` delayable. The pedestrian is ensured to survive if the bus driver is infinitely patient.

Case 3: Now, suppose the driver's patience wears out and he must pass before a definite upper time bound 2. The pedestrian must jump after the first clock tick and before the patience of the bus driver runs out (upon the second clock tick). The largest controllable and SANE DESP of G is empty if u_{jump} is increased to 2.

Case 4: Supposed that the pedestrian removes his headset and regains his hearing and can be told to jump if his life is threatened. The bus driver is willing to stop for up to a certain amount of time. The bus will stop until the second clock tick has lapsed and the pedestrian must jump after the first tick and before the second tick. With *jump* being made expeditible, the pedestrian's survival is guaranteed even though u_{jump} has been increase to ∞.

References

[1] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems," *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems*, Vol. 77(1), pp. 81-98, 1989.

[2] H. Wong-Toi and G. Hoffmann, "The control of dense real-time discrete event systems," *Prof. 30th Conf. Decision and Contr.* pp. 1527-1528, Dec. 1991.

[3] J.S. Ostroff, "Formal methods for the specification and design of real-time safety critical systems," *Journal of Systems and Software.* April, 1992.

[4] I. Brave and M. Heymann, "Formulation and control of real-time discrete event systems," *Proc. 27nd Conf. Decision Contr.*, pp. 1133-1134, Dec. 1988.

[5] J.S. Ostroff, *Temporal logic for real-time systems.* John Wiley and Sons, 1989.

[6] J.S. Ostroff and W.M. Wonham "A framework for real-time discrete event control," *IEEE Trans. on Automatic Control.* Vol. 35(4), April, 1990.

[7] C.H. Golaszewski and P.J. Ramadge, "Control of discrete event processes with forced events," *Proc. 26nd Conf. Decision Contr.*, pp. 247-251, Dec. 1987.

[8] C.H. Golaszewski and P.J. Ramadge, "On the control of real-time discrete event systems," *Proc. 23rd Conf. on Information Systems and Signals.*, pp. 98-102, Princeton, NJ, Mar. 1989.

[9] J.S. Ostroff, "Synthesis of controllers for real-time discrete event systems," *Proc. 28nd Conf. Decision Contr.*, pp. 138-144, Dec. 1989.

[10] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[11] K. Rudie, "Software for the control of discrete event systems," Master's thesis, Dept. Elec. Eng., Univ. Toronto, Feb. 1988.

[12] S.D. O'Young, "TTM_talk: a user's guide," *Systems Control Group Rpt.*, #9202, Dept. Elec. Eng., Univ. Toronto, 1992.

[13] B.A. Brandin and W.M. Wonham, "The supervisory control of timed discrete event systems," *Systems Control Group Rpt.*, #9112, Dept. Elec. Eng., Univ. Toronto, 1991.

[14] B.A. Brandin, W.M. Wonham and B. Benhabib, "Manufacturing cell supervisory control – a timed discrete event systems approach," *Proc. IEEE International Conf. on Robotics and Automaton*, Nice, France, May, 1992.

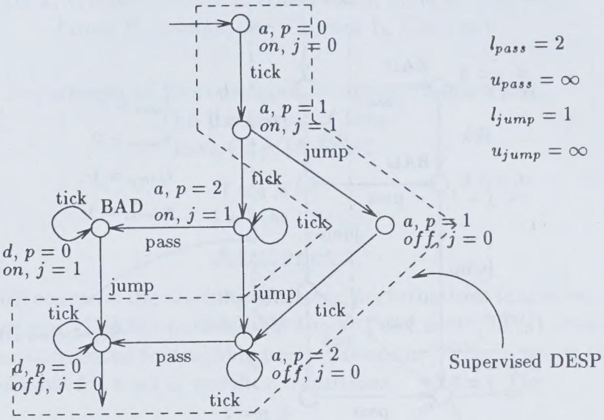
[15] C.A.R. Hoare, *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[16] S.D. O'Young, "On the synthesis of supervisors for timed discrete event processes," *Systems Control Group Rpt.*, #9107, Dept. Elec. Eng., Univ. Toronto, 1991.

Appendices

Each state is coded with the symbols a, d, on, and off representing the activity assignments `approach`, `depart`, `onRoad` and `offRoad` respectively, and the symbols p and j representing the local timers t_{pass} and t_{jump}, respectively. The trivial timing constraints on events σ where $u_\sigma = \infty$ and $l_\sigma = 0$ are not listed.

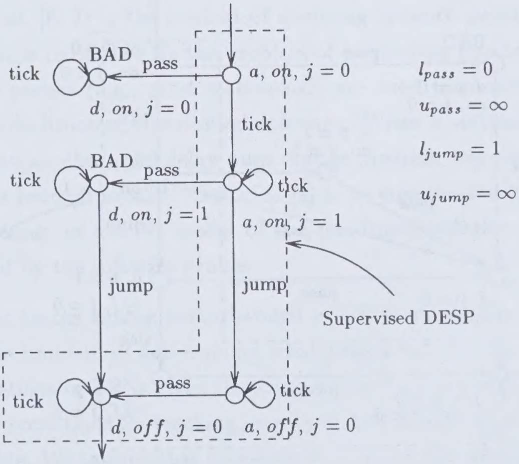
Appendix A



a, $p = 0$
on, $j = 0$
tick
a, $p = 1$
on, $j = 1$
jump
tick
a, $p = 2$
on, $j = 1$
tick
BAD
tick
pass
d, $p = 0$
on, $j = 1$
jump
jump
a, $p = 1$
off, $j = 0$
tick
tick
pass
a, $p = 2$
off, $j = 0$
d, $p = 0$
off, $j = 0$
tick
Supervised DESP
$l_{pass} = 2$
$u_{pass} = \infty$
$l_{jump} = 1$
$u_{jump} = \infty$

Fig. A.1

Appendix B



BAD
tick
pass
a, on, $j = 0$
d, on, $j = 0$
tick
BAD
tick
pass
tick
d, on, $j = 1$
a, on, $j = 1$
jump
jump
Supervised DESP
tick
pass
tick
d, off, $j = 0$
a, off, $j = 0$
$l_{pass} = 0$
$u_{pass} = \infty$
$l_{jump} = 1$
$u_{jump} = \infty$

Fig. B.1

Appendix C

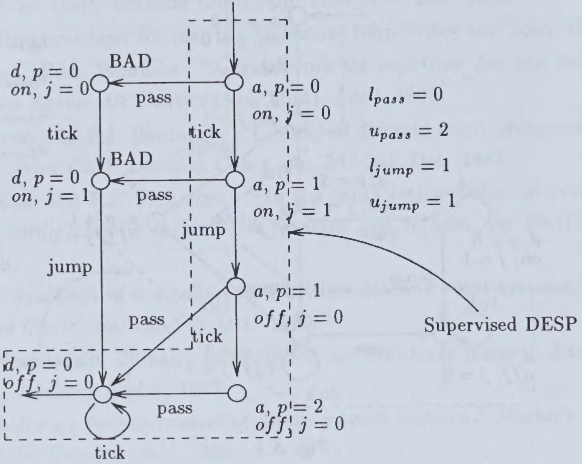

BAD
$d, p = 0$
$on, j = 0$
pass
$a, p = 0$
$on, j = 0$
$l_{pass} = 0$
$u_{pass} = 2$
$l_{jump} = 1$
$u_{jump} = 1$
tick
tick
BAD
$d, p = 0$
$on, j = 1$
pass
$a, p = 1$
$on, j = 1$
jump
jump
$a, p = 1$
$off, j = 0$
pass
tick
Supervised DESP
$d, p = 0$
$off, j = 0$
pass
$a, p = 2$
$off, j = 0$
tick

Fig. C.1

Appendix D

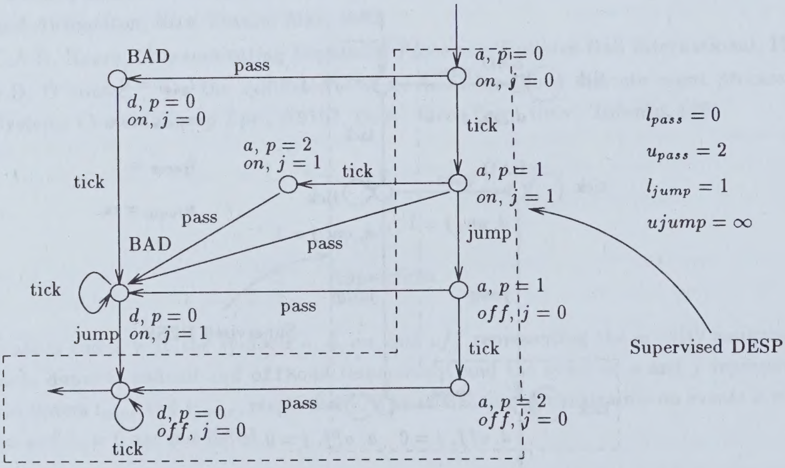

BAD
pass
$a, p = 0$
$on, j = 0$
$d, p = 0$
$on, j = 0$
tick
$l_{pass} = 0$
$u_{pass} = 2$
$l_{jump} = 1$
$ujump = \infty$
$a, p = 2$
$on, j = 1$
tick
$a, p = 1$
$on, j = 1$
tick
pass
BAD
pass
jump
tick
$a, p = 1$
$off, j = 0$
pass
$d, p = 0$
$on, j = 1$
jump
tick
Supervised DESP
pass
$d, p = 0$
$off, j = 0$
$a, p = 2$
$off, j = 0$
tick

Fig. D.1

Necessary Conditions for Tracking Timing Perturbations In Timed Petri Nets*

Kyle J. Williams, Mark S. Andersland, Julie A. Gannon,
James E. Lumpp, Jr., Thomas L. Casavant

Department of Electrical and Computer Engineering
The University of Iowa
Iowa City, IA 52242

Abstract

This work concerns the tracking of timing perturbations in discrete event dynamic systems (DEDS) modeled by timed Petri nets (TPN). Necessary conditions are presented for tracking perturbations in TPNs in which timings are only available for a set of monitor transitions.

1. INTRODUCTION

This work concerns the tracking of timing perturbations in discrete event dynamic systems (DEDS) modeled by timed Petri nets (TPN). Analysis of this sort was first considered by Ho et. al. [6, 7] in the context of queueing network simulation and optimization. Our interest is motivated by the problem of accounting for the intrusion that arises when software probes (e.g., print statements) are used to monitor the internal state of executing, asynchronous, distributed software. When a software probe delays one process relative to another, the delay may change program behavior to the point that the observations become invalid. Our idea [1] is to compensate for the monitoring intrusion by tracking, in a TPN model of the monitored software [8], the timing perturbations induced by the software probes.

An algorithm that tracks timing perturbations in TPNs is described in [4]. In this paper we consider the problem of determining what timing information is necessary to track timing perturbations in TPNs given the firing times of a set of monitor transitions. To identify necessary conditions for tracking, one must first specify what a priori timing information is available. We assume that no information about the relative TPN timings

---

*This research supported by DARPA Grant N00174-91-C-0116, the Hewlett Packard Faculty Development Program, and NSF Grant Number CCR-9024484.

is available. Instead, only weak general properties of the TPN are assumed known. This is a worst-case assumption in the sense that it leads to the strongest necessary conditions.

2. NOTATION

For the purposes of our study, a TPN is a bipartite, directed graph of transitions, places, and arcs, denoted by the 4-tuple $\mathcal{P} := (T, P, A, x_0)$, where

T is a finite set of transitions;

P is a finite set of places;

$A \subset (P \times T) \cup (T \times P)$ is a set of directed arcs;

x_0 is a function $x_0 : P \to \{0, 1, \ldots\}$ returning the initial number of tokens in place p.

The TPN is ordinary, first-in first-out (FIFO), and has places with unbounded capacity. The net's timed behavior is determined by the net's *firing delay schedule* $V := \{v_t : t \in T\}$, $v_t = \{v_t(1), v_t(2), \ldots\}$. Here $v_t(k) \in I\!R^+$, $k \in I\!N$, denotes the firing delay incurred by the kth firing of transition t.[1] The normal firing rules apply [2]. Although V may be random, the behavior of the net is deterministic for fixed V. In particular, this implies that the behavior of tokens at decision places[2] is deterministic. For example, nets with no decision places [5] and nets with decision places that are controlled by exogenous decision sequences [2] are deterministic. In essence, fixing V imposes a unique total ordering on the firing times of transitions that interact. Hence, V completely characterizes the behavior of the system. To differentiate between the two behaviors, we denote the firing delay schedules of the unperturbed and perturbed nets by $\overline{V}$ and V, respectively. In the sequel, quantities in the unperturbed net will be indicated by overbars.

We use, for $t \in T$ and $k \in I\!N$, the following notation:

$\mathcal{I}(u)$	is the set of input places or transitions to transition or place u, respectively, e.g., $\mathcal{I}(t) = \{p \in P : \exists\, (p, t) \in A\}$.
(t, k)	denotes the kth firing of t.
$\overline{\tau}_t(k), \tau_t(k)$	is the *firing time* of (t, k), i.e., the time that t completes firing for the kth time in the unperturbed and perturbed net, respectively. $\overline{\tau}_t$ and τ_t are the sequences of these times for t.
$\Delta_t(k)$	$:= v_t(k) - \overline{v}_t(k)$ is the *perturbation* induced on the kth firing of t. Δ_t is the sequence of these perturbations for t.
$M \subset T$	is the set of *monitor* transitions, i.e., transitions t for which $\overline{\tau}_t$ are known and $\Delta_t \neq 0$.

---

[1] Note that the firing schedule V is different from the clock structure defined in [3].

[2] A decision place is a place with more than one output transition.

Because the behavior of the net is deterministic for fixed V, we can also define the following notation:

$C_V(t,k)$ $:= \{(s_i, h_i) : s_i \in \mathcal{I}(\mathcal{I}(t)),\ h_i \in I\!N$, and (s_i, h_i) produces, under V, a token consumed by $(t,k)\}$. In general, $\mid C_V(t,k) \mid \leq \mid \mathcal{I}(t) \mid$ because (t,k) may be consuming a token that was part of the initial marking.

$I_V(t,k;u,\ell)$ $:= \tau_u(\ell) - \tau_t(k) - v_u(\ell)$, $(t,k) \in C_V(u,\ell)$, is the *idle time* under V that the token produced by (t,k) incurs at an intervening place before it enables (u,ℓ).

Using the characterization of "upstream" provided by $C_V(\cdot,\cdot)$ we define, for fixed V, a *token track* of length n to be a sequence of transition firings $\{(s_i,h_i)\}_{i=1}^n$, such that $(s_{i-1}, h_{i-1}) \in C_V(s_i, h_i)$, for $i = 2, 3, \ldots, n$. Token tracks are snapshots of TPN behavior analogous to the paths of the *Evolution Tree* defined in [2] for places. A token track $\{(s_i,h_i)\}_{i=1}^n$ is said to be *rooted* when $s_j \notin M$ for all $j = 2, 3, \ldots, n-1$, and (s_1, h_1) satisfies at least one of the following conditions: (a) $s_1 \in M$; (b) $\mid \mathcal{I}(s_1) \mid = 0$; or (c) $\mid \mathcal{I}(s_1) \mid = 1$ and $x_0(\mathcal{I}(s_1)) \geq h_1$.

We define $D_{\mathcal{P}}$ to be the set of all possible token tracks that exist in TPN $\mathcal{P}$ under some V. Because every transition that fires belongs to a token track of length one, some token track in $D_{\mathcal{P}}$ passes through every L1-live transition [9]. When $D_{\mathcal{P}}$ is empty, no transition fires under any V; hence $D_{\mathcal{P}}$ is non-empty for all systems of interest.

3. PROBLEM STATEMENT

To develop necessary conditions for tracking perturbations in TPNs, we make the following assumptions:

A1. Firing delays $\overline{v}_t(k)$ and $v_t(k)$, $t \in T \setminus M$, $k \in I\!N$, only depend on t's firing index k, not on t's enabling time.

A2. Token tracks do not change under perturbation, i.e., for any token track $S \in D_{\mathcal{P}}$ that exists under $\overline{V}$, S exists under V.

A3. For any token track $S = \{(s_i,h_i)\}_{i=1}^n \in D_{\mathcal{P}}$ and index set $\overline{B} \subset \{2, 3, \ldots, n\}$, there exists a $\overline{V}$ such that S is a token track under $\overline{V}$ and $\overline{I}_{\overline{V}}(s_{j-1}, h_{j-1}; s_j, h_j) = 0$ for all $j \in \overline{B}$, and $\overline{I}_{\overline{V}}(s_{k-1}, h_{k-1}; s_k, h_k) \neq 0$ for all $j \notin \overline{B}$.

A4. A3 holds with $\overline{B}$ and $\overline{V}$ replaced by B and V, respectively.

Assumption *A1* ensures that the non-monitor transitions are "time-invariant", i.e., that $\overline{v}_t(k) = v_t(k)$, $t \in T \setminus M$, $k \in I\!N$. When timing perturbations alter token tracks, tracking the perturbations becomes more difficult. *A2* ensures that, aside from changes in transition firing times, each transition produces and consumes the same tokens under perturbation. This assumption is not overly restrictive because it always applies to some subtrack of every token track. *A3* and *A4* imply that no information about relative TPN timings is available. These are the worst-case assumptions in that they lead to the strongest necessary conditions. When timing information can be derived from net structure, the necessary conditions may be satisfied a priori (e.g., if a net is dead under all V, all timings are derivable from structure). Note that we do not assume that the firing delay schedule V is known; all timing information necessary for tracking perturbations must be extracted from the net structure and the firing times $\overline{\tau}_t$, $t \in M$.

The 0th order perturbation tracking problem (tracking up to a token track change) can be stated as follows.

(TP0:) Given a TPN satisfying assumptions *A1* - *A4*, given that the firing times $\overline{\tau}_t$ of the monitor transitions $t \in M$ are known, and given that the firing durations $\overline{v}_t$ of these transitions are perturbed, find the sequence of perturbed firing times τ_t of the transitions $t \in M$.

4. PRELIMINARY RESULTS

Let $\{(s_i, h_i)\}_{i=1}^{n}$ be a token track of length n in the unperturbed net. Because $\overline{\tau}_{s_n}(h_n)$ can be expressed as a telescoping sum along this track

$$\begin{aligned} \overline{\tau}_{s_n}(h_n) &= \overline{\tau}_{s_1}(h_1) + \sum_{i=2}^{n}(\overline{\tau}_{s_i}(h_i) - \overline{\tau}_{s_{i-1}}(h_{i-1}) - \overline{v}_{s_i}(h_i)) + \sum_{i=2}^{n}\overline{v}_{s_i}(h_i) \\ &= \overline{\tau}_{s_1}(h_1) + \sum_{i=2}^{n}\overline{I}_{\overline{V}}(s_{i-1}, h_{i-1}; s_i, h_i) + \sum_{i=2}^{n}\overline{v}_{s_i}(h_i). \end{aligned} \tag{1}$$

Because *A2* ensures that the token track is the same in the perturbed net, we can also write

$$\tau_{s_n}(h_n) = \tau_{s_1}(h_1) + \sum_{i=2}^{n} I_V(s_{i-1}, h_{i-1}; s_i, h_i) + \sum_{i=2}^{n} v_{s_i}(h_i). \tag{2}$$

Subtracting (2) from (1) and using the definition of $\Delta_{s_i}(h_i)$, we find

$$\begin{aligned} \overline{\tau}_{s_n}(h_n) - \tau_{s_n}(h_n) &= \overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1) + \sum_{i=2}^{n}(\overline{I}_{\overline{V}}(s_{i-1}, h_{i-1}; s_i, h_i) \\ &\qquad - I_V(s_{i-1}, h_{i-1}; s_i, h_i)) - \sum_{i=2}^{n}\Delta_{s_i}(h_i). \end{aligned} \tag{3}$$

When $\{(s_i, h_i)\}_{i=1}^{n}$ is a rooted token track, by definition, $\Delta_{s_i}(h_i) = 0$, for $i = 2, 3, \ldots, n-1$; hence

$$\begin{aligned} \overline{\tau}_{s_n}(h_n) - \tau_{s_n}(h_n) &= \overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1) + \sum_{i=2}^{n}(\overline{I}_{\overline{V}}(s_{i-1}, h_{i-1}; s_i, h_i) \\ &\qquad - I_V(s_{i-1}, h_{i-1}; s_i, h_i)) - \Delta_{s_n}(h_n). \end{aligned} \tag{4}$$

Lemma 1: *Fix V, and suppose that $\{(s_i, h_i)\}_{i=1}^{n}$ is a rooted token track under V. Assuming that τ_t is known for all $t \in M$, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)$ is known.*

Proof: By definition, (s_1, h_1) must satisfy at least one of the three rooted token track conditions. It suffices to consider these cases separately.

Case 1: When $s_1 \in M$, $\overline{\tau}_{s_1}(h_1)$, $h_1 \in I\!N$ is known by definition, and $\tau_{s_1}(h_1)$, $h_1 \in I\!N$, is known by the assumption of the Lemma. Hence, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)$ is known.

Case 2: Suppose $| \mathcal{I}(s_1) |= 0$. When $s_1 \in M$, then as in Case 1, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)$ is known. Suppose $s_1 \notin M$. Because transition s_1 is always enabled,

$$\overline{\tau}_{s_1}(h_1) = \sum_{j=1}^{h_1} \overline{v}_{s_1}(j) \tag{5}$$

$h_1 \in I\!N$. By the definitions of $\Delta_{s_1}(h_1)$ and M, in the perturbed net $\Delta_{s_1}(h_1) = 0$, for all $h_1 \in I\!N$, and (5) becomes

$$\tau_{s_1}(h_1) = \sum_{j=1}^{h_1} v_{s_1}(j) = \sum_{j=1}^{h_1} (\overline{v}_{s_1}(j) + \Delta_{s_1}(j)) = \sum_{j=1}^{h_1} \overline{v}_{s_1}(j). \tag{6}$$

Subtracting (6) from (5) yields

$$\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1) = \sum_{j=1}^{h_1} \overline{v}_{s_1}(h_1) - \sum_{j=1}^{h_1} \overline{v}_{s_1}(h_1) = 0. \tag{7}$$

Hence, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)$ is known.

Case 3: When $| \mathcal{I}(s_1) |= 1$, $x_0(\mathcal{I}(s_1)) > h_1$, s_1 has been constantly enabled by initial tokens because the net is FIFO. Hence, as in Case 2, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1) = 0$ is known. ■

5. NECESSARY CONDITIONS

Theorem 1: *To solve perturbation tracking problem TP0, it is necessary to know*

N1: the perturbation $\Delta_{s_n}(h_n)$, for all rooted token tracks $S = \{(s_i, h_i)\}_{i=1}^{n} \in D_{\mathcal{P}}$, $s_n \in M$.

N2: the idle times $I_{\overline{V}}(s_{b-1}, h_{b-1}; s_b, h_b)$, $I_V(s_{b-1}, h_{b-1}; s_b, h_b)$, $b = 2, 3, \ldots, n$, for all rooted token tracks $S = \{(s_i, h_i)\}_{i=1}^{n} \in D_{\mathcal{P}}$, $s_n \in M$, $n > 1$.

Proof: Assume that the solution to TP0 is known, i.e., $\tau_u(\ell)$ is known for all $u \in M$, $\ell \in I\!N$.

N1: Fix the rooted token track $S = \{(s_i, h_i)\}_{i=1}^{n} \in D_{\mathcal{P}}$, $s_n \in M$. It suffices to demonstrate the existence of a V and $\overline{V}$ such that, under assumptions *A1 - A4* and the hypothesis, $\Delta_{s_n}(h_n)$ is known.

By *A2* - *A4*, there exists a V and a $\overline{V}$ such that S exists under V and $\overline{V}$ and $\overline{B} = B = \{2,3,\ldots,n\}$, i.e., $\overline{I}_{\overline{V}}(s_{i-1},h_{i-1};s_i,h_i) = I_V(s_{i-1},h_{i-1};s_i,h_i) = 0$, for all $i \in \{2,3,\ldots,n\}$. Substitution of these idle times in (4) yields, after rearrangement:

$$\overline{\tau}_{s_n}(h_n) - \tau_{s_n}(h_n) - (\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)) = -\Delta_{s_n}(h_n). \tag{8}$$

$\overline{\tau}_{s_n}(h_n)$ is known by the definition of M, $\tau_{s_n}(h_n)$ is known by hypothesis, and $(\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1))$ is known by Lemma 1. Hence, $\Delta_{s_n}(h_n)$ is known.

N2: Fix the rooted token track $S = \{(s_i,h_i)\}_{i=1}^n \in D_{\mathcal{P}}$, $s_n \in M$, $n > 1$. It suffices to demonstrate the existence of a $\overline{V}$ and V such that, under assumptions *A1* - *A4* and the hypothesis, $\overline{I}_{\overline{V}}(s_{b-1},h_{b-1};s_b,h_b)$ and $I_V(s_{b-1},h_{b-1};s_b,h_b)$ are known for all $b = 2,3,\ldots,n$.

Fix $b \in \{2,3,\ldots,n\}$. By *A3* there exists a $\overline{V}$ such that S exists under $\overline{V}$ and $\overline{B} = \{2,3,\ldots,b-1,b+1,\ldots,n\}$, i.e., $\overline{I}_{\overline{V}}(s_{i-1},h_{i-1};s_i,h_i) = 0$, for $i \in \{2,3,\ldots,b-1,b+1,\ldots,n\}$. By *A2* and *A4*, there exists a V such that S exists under V and $B = \{2,3,\ldots,n\}$, i.e., $I_V(s_{i-1},h_{i-1};s_i,h_i) = 0$, for $i \in \{2,3,\ldots,n\}$. Substituting these idle times into (4), we find that

$$\overline{\tau}_{s_n}(h_n) - \tau_{s_n}(h_n) = \overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1) + \overline{I}_{\overline{V}}(s_{b-1},h_{b-1};s_b,h_b) - \Delta_{s_n}(h_n). \tag{9}$$

$\overline{\tau}_{s_n}(h_n)$ is known by the definition of M, $\tau_{s_n}(h_n)$ is known by hypothesis, $\overline{\tau}_{s_1}(h_1) - \tau_{s_1}(h_1)$ is known by Lemma 1, and $\Delta_{s_n}(h_n)$ must be known by N1. Hence, $\overline{I}_{\overline{V}}(s_{b-1},h_{b-1};s_b,h_b)$ is known, for all $b = 2,3,\ldots,n$, for any rooted token track $\{(s_i,h_i)\}_{i=1}^n$, with $n > 1$.

Similarly, interchanging the role of $\overline{I}_{\overline{V}}(s_{b-1},h_{b-1};s_b,h_b)$ and $I_V(s_{b-1},h_{b-1};s_b,h_b)$ we see that $I_V(s_{b-1},h_{b-1};s_b,h_b)$ is known, for all $b = 2,3,\ldots,n$, for any rooted token track $\{(s_i,h_i)\}_{i=1}^n$, $n > 1$. ■

6. DISCUSSION

Theorem 1 characterizes the timing information required to solve TP0. In essence, it states that we have to know the durations of the perturbations incurred by the firing delays of the monitor transitions, and that we have to be able to track perturbations when they interact. To track perturbations we must, in turn, know the idle times of tokens in the net.

Theorem 1 does not specify how we should acquire this information. When this information can not be derived from net structure we are forced to derive it from the monitor transition firing times. This is discouraging when tokens can idle in any place (e.g., in TPN models of queueing networks) because it implies that every transition firing time must be derivable from monitor firing times. For many important systems this is not the case. Net structure imposes constraints on the locations of non-zero idle times (i.e., some idle times are known to be zero for all possible firing delay schedules V), and consequently, many transitions need not be monitored.

For instance, in the TPN models of the distributed message passing systems that motivated this work, non-zero idle times can only arise at places upstream from the transitions that synchronize the message passing. Accordingly, it is only necessary (and sufficient—see [4]) to monitor transitions in the vicinity of these synchronizing transitions.

REFERENCES

[1] M. S. Andersland and T. L. Casavant, "Recovering uncorrupted event traces from corrupted event traces in parallel/distributed computing systems," *Proc. of the 1991 International Conference on Parallel Processing*, August 1991, pp. II.108-II.116.

[2] F. Baccelli, G. Cohen, and B. Gaujal, "Evolution equations of timed Petri nets," *Proc. of the 30th IEEE Conference on Decision and Control*, Vol. 2, December 1991, pp. 1139-1144.

[3] C. G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*, Irwin, Inc., and Aksen Associates Inc., 1993.

[4] J. A. Gannon, J. E. Lumpp, Jr., K. J. Williams, M. S. Andersland, and T. L. Casavant, "Tracking timing perturbations in timed Petri nets," *University of Iowa Electrical and Computer Engineering Technical Report* TR-ECE-920304, March 1992.

[5] H. P. Hillion and J.-M. Proth, "Performance evaluation of job-shop systems using timed event-graphs," *IEEE Transactions on Automatic Control*, Vol. 34, No. 1, January 1989, pp. 3-9.

[6] Y.-C. Ho, X.-R. Cao, and C. Cassandras, "Infinitesimal and finite perturbation analysis for queueing networks," *Automatica*, Vol. 19, No. 4, April 1983, pp. 439-445.

[7] Y.-C. Ho and X.-R. Cao, *Perturbation Analysis of Discrete Event Dynamic Systems*, Kluwer Academic Publishers, 1991.

[8] J. E. Lumpp, Jr., J. A. Gannon, M. S. Andersland, and T. L. Casavant, "A technique for recovering from software instrumentation intrusion in message-passing systems," *University of Iowa Electrical and Computer Engineering Technical Report* TR-ECE-920817, August 1992.

[9] T. Murata, "Petri nets: properties, analysis, and applications," *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989, pp. 541-580.

A Framework for Timed Discrete-Event Systems

Tan-Jan Ho and W. M. Wonham

Systems Control Group
Department of Electrical Engineering
University of Toronto
Toronto, M5S 1A4
CANADA

Summary

In this paper, we present a new Hybrid Timed Model (HTM) by combining the modeling features of TTM's (Timed Transition Models) [3] and Timed Petri Nets.

An HTM is a 10-tuple $G_{tim} = (\Sigma_{tim}, \mathsf{X}_{tim}, BE, TF, A, B, C, STI, \delta_{tim}, \mathsf{X}_{o,tim})$, described as follows.

- Σ_{tim} is a finite, nonempty alphabet of events. We partition $\Sigma_{tim} := \Sigma_{unc} \cup \Sigma_{con}$ where the disjoint subsets Σ_{unc} and Σ_{con} comprise the uncontrollable events and controllable events by analogy with RW theory [6]. We assume the global 'tick' $\in \Sigma_{con}$ and bring in the forcible event set $\Sigma_f \subseteq \Sigma$ where $\Sigma := \Sigma_{tim} - \{\text{tick}\}$.
- X_{tim} denotes the triple (X_a, X_d, TV) where X_a and X_d represent the activity and data vector respectively. TV is the time bound vector for all non-tick events. Hereafter, X denotes the pair (X_a, X_d).
- BE is a finite set of Boolean-valued functions $e : \mathsf{X}_{tim} \to \{0, 1\}$.
- TF is a finite set of activity and data vector transformation functions $h : \mathsf{X} \to \mathsf{X}$.
- A is a Boolean mapping, $A : (\Sigma \times BE) \times \mathsf{X}_{tim} \to \{0, 1\}$ with $A(\sigma, e, X_{tim}) := e(X_{tim})$.
- B is a mapping, $B : BE \to TF$.
- C is a mapping, $C : \Sigma \times BE \to \Sigma \times TF$ with $(\sigma, e) \longmapsto (\sigma, B(e))$.
- STI is a time interval mapping, $STI : \Sigma \times BE \to (\mathbb{Z}_+ \cup \{\infty\}) \times (\mathbb{Z}_+ \cup \{\infty\})$, where $\mathbb{Z}_+ := \{0, 1, 2, ...\}$. $STI((\sigma, e)) = (l, u)$. l is the static lower time bound, the delay time of an event, and u is the static upper time bound, corresponding to the timeout of an event. If $u \neq \infty$, $l \leq u$; otherwise, $l < u$.
- δ_{tim} is a transition function, $\delta_{tim} : \Sigma_{tim} \times \mathsf{X}_{tim} \to \mathsf{X}_{tim}$.
- $\mathsf{X}_{o,tim}$ is the initial state set.

The semantics of HTMs follows that of TTM's. In our framework, we adopt state-feedback control . Also, in accord with [1], we impose the criterion that control should at most restrict the uncontrolled behavior of a plant and never enlarge it. Because

of the presence of timing information for each non-tick event, we consider that the control technology not only employs the enablement and disablement strategy used in [6], but also includes a forcing mechanism. A forcible event is one that can preempt the global 'tick'. If, at a state in the plant, 'tick' is defined and one or more forcible events are eligible to occur immediately , then our forcing technology permits the effective erasure of 'tick' from the current list of defined events, namely guarantees that the plant takes action before a global 'tick' occurs. Based on the structure of HTMs and assumptions in our set-up, we develop a systematic design procedure for constructing a timed state-feedback controller along the lines of RW modular logic feedback theory [5]. The representations of the plant and the closed-loop timed system are both HTMs.

The practicability of our HTMs and design procedure can be demonstrated by the following example used in [2]. Suppose a nuclear plant contains two rods , ROD1 and ROD2. The normal operations of ROD1 are described as follows. 1. ROD1 is lowered by the controllable transition Lower1$[5,\infty]$ from the activity 'ready' to 'down' and the data variable c_1 is incremented by one. 2. ROD1 is raised by the uncontrollable transition Up1$[7,20]$ from the activity 'down' to 'up'. 3. If $c_1 > 0$, then ROD1 can be serviced from the activity 'up' to 'repair' via the controllable and forcible transition Service1$[0,\infty]$ and c_1 is reset to zero. 4. ROD1 returns from the activity 'repair' to 'up' via the uncontrollable transition Reset$[0,\infty]$ shared with ROD2. 5. ROD1 moves from the activity 'up' to 'ready' by the uncontrollable transition Push1$[3,\infty]$. The operations of ROD2 are described as the same as those of ROD1. The control problem is to guarantee that the operations of ROD1 and ROD2 satisfy $c_1 \leq 10$, $c_2 \leq 10$, and $c_1 + c_2 \leq 15$ in a minimally restrictive way. In [2], Ostroff proposed a heuristic approach and a solution for this problem, whereas based on our framework and systematic design algorithm, we obtain a less conservative solution compared with the solution given in [2]. Also, in comparison with [1] and [4], our method efficiently exploits arithmetic properties of data variables (here c_1 and c_2) when these are present.

References

[1] B.A. Brandin and W.M. Wonham. Supervisory control of timed discrete-event system. Technical Report 9210, Systems Control Group, Department of Electrical Engineering, University of Toronto, August 1992.

[2] J.S. Ostroff. Systematic development of real-time controllers. In *Proceedings of the ECC 91 European Control Conference*, pages 522–533, Paris, France, 1991.

[3] J.S. Ostroff and W.M. Wonham. A framework for real-time discrete-event control. *IEEE Transactions on Automatic Control*, 35:386–397, 1990.

[4] S. O'Young. On the synthesis of supervisors for timed transition models. Technical Report 9107, Systems Control Group, Department of Electrical Engineering, University of Toronto, November 1991.

[5] P.J. Ramadge and W.M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control and Optimization*, 25:1202–1218, 1987.

[6] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25:206–229, 1987.

ASSIGNMENT OF 2^k TRUNK GROUPS IN MULTICHANNEL SWITCHES USING GENERALIZED BINARY ADDRESSES

H. Saidi, P.S. Min
Washington University, Campus Box 1127
St. Louis, MO 63130

M.V. Hegde
Louisiana State University
Baton Rouge, LA 70803

1 Introduction

In order to alleviate the processing speed constraint of electronic switches in broadband networks, multichannel switches have been proposed and studied in the literature [1] [2] [3] [4]. Multichannel switches can support the extremely high transmission rates (e.g., terabits per second) of lightwave technology by introducing the concept of trunk grouping. Instead of being routed to a specific output link, a packet is routed to any link belonging to appropriate trunk group. In reality, a trunk group may correspond to a single transmission fiber operating at a terabit rate, and multichannel switches allow a number of users to share the bandwidth by dynamically allocating time slots. The advantages of statistically sharing a higher channel capacity are well known [7]. For example, it increases the link efficiency by reducing the burstiness in the arrival pattern of incoming packets.

Recent papers [5] [6] attempt to combine a self-routing network with multichannel switches in order to reduce the complexity in call set-up control. In [5] and [6], all packets belonging to a trunk group are given a single binary address, and the routing within the switch is performed based only on this address as described in detail in Section 2. We note that while these methods involve simple call set up control for high throughput rates, their performance is sensitive to the *a priori* destination or trunk group probability of input packets. To utilize the advantage of the methods, the assignment of trunk groups needs to be updated whenever the *a priori* probability changes. Since the assignment process involves calculating the probability of different packets at each output link, a frequent update in the trunk group assignment may be difficult.

This paper is aimed at formulating a method of trunk group assignment that is simple and implementable in real time. The main idea of this paper is based on a generalized addressing scheme that assigns an arbitrary number of trunk groups in multichannel switches. For mathematical convenience, we assume in this paper that different trunk groups are requested with equal probability by each incoming packet. This is a rather restrictive assumption, and generalization to arbitrary probabilities is a topic of continuing study, the results of which will be reported in a subsequent paper.

Under the proposed addressing scheme, the output links of a switch can be partitioned into 2^k trunk groups using only the last k bits of the link address. As the number of trunk groups changes, the optimal assignment of output links can be made in real time with a minimal amount of computation. Unlike the method in [5], the assignment is independent of the level of offered load to the switch.

The organization of this paper is as follows: In Section 2, we review the multichannel switch used by [5]. In Section 3, we illustrate the main idea of this paper using an example with two trunk groups. In Section 4, we expand the idea in Section 3 to 2^k trunk groups. The conclusions of this paper are given in Section 5.

2 Overview of the Statistical Data Fork

In [5], the author presented a multichannel switch called the *Statistical Data Fork (SDF)*. The SDF is a multichannel switch based on the Omega network, and employs no internal buffer and no upfront call set up process. By utilizing a self-routing network, the SDF improves the processing speed and hence the throughput. On arrival, each packet is simply given a code word which identifies the trunk group that the packet belongs to, and based on this code word, the packet is routed without intervention from the central controller of the switch. (Packets routed to the same output trunk group are said to be of the same type.) Detailed operations of the SDF are described below. We note that a similar type of switch was presented in [6] which however, is different from that of [5] in that it employs internal buffers. The authors called this switch the *Buffered Statistical Data Fork (BSDF)*.

Let $N = 2^n$ be the number of input links in the SDF. Then, the SDF consists of n interconnected stages of the Omega network wherein each stage contains $N/2$ components. These components perform switching functions between two inputs and two outputs, and we refer to them as switching elements (rather than the switch itself). We label the stages between 1 and n starting from the left, (see Figure 1.) and at each stage, label the switching elements between 0 and $(N/2 - 1)$ starting from the top. Let S_{lm} denote switch l located at stage m. The input links to the SDF are labeled between 0 to $N - 1$ starting from top. The output links are labeled similarly. We will sometimes refer to these link labels as the *addresses* for the links. An example of the SDF is shown in Figure 1.

At the beginning of a time slot, packets enter the SDF through the input links. As they enter, each packet is given an n bit binary code word which uniquely identifies a trunk group. Within the SDF, packets traverse through n stages of 2×2 switching elements, and at each stage, the packets must exit through one of the two outputs since there is no internal buffer.

The n bits contained in a code word determine the *preference* of the packet. The preference of the packet at stage i is determined by the i^{th} bit of the code word given to the packet; if the i^{th} bit is 0 then the packet prefers to exit through the upper output (or output 0) and if it is 1, it prefers to exit through the lower output (or output 1).

It is possible that two packets arrive simultaneously at the two input links of a switch. Then, the output that the packet actually exits on depends on the other packet in the same switching element at the same time. If the preferences of the two packets are different, each packet exits through the preferred output according to the code word. On the other hand, a contention can take place if the packets have the same preference. In this case, the priority between the two packets must resolve the contention. When they both have the same priority, the choice is made arbitrarily.

Assuming that there are two trunk groups, consider two code words $000\cdots00$ and $111\cdots11$ with maximum Hamming distance. The first packet group (with the code word $000\cdots00$) has preference 0 at all stages, and the second packet group (with the code word $111\cdots11$) has preference 1.

Whenever a contention occurs, one of the packets is deflected to the output that is not preferred. The basic idea of the SDF is that by assigning individual output links to a proper trunk group, the probability of packets that are routed incorrectly (i.e., to the wrong trunk group) is minimized. The SDF accomplishes this by explicitly calculating at each output link the probabilities that different packet types are routed to that link. Then, the optimal assignment of this output link is to the trunk group whose packet type has the highest probability.

For any link (i.e., input, output, or internal link) in the SDF, let $(p, \tilde{p})$ denote that the probability of the first packet type is p and the second type is $\tilde{p}$ where $p + \tilde{p} \leq 1$. We call $(p, \tilde{p})$ the type probabilities of this link. Utilization of a link is measured by $p + \tilde{p}$, which is allowed to be less than 1 in in order to account for partial utilization.

Let (p_0, p_1) and (q_0, q_1) be the type probabilities for the two input links of a 2×2 switching element. Then, $(p_0 + q_0 - p_0q_0, p_1q_1)$ is the probability for output zero and $(p_0q_0, p_1 + q_1 - p_1q_1)$ for output one. Since the outputs of this switching element become the inputs to others in the next stage, we can calculate the probabilities of all links in the SDF by repeating this calculation.

Assume that for an output link of the SDF, the type probability is determined to be (r_0, r_1). Assuming no further constraint, this link is assigned to the first trunk group if $r_0 > r_1$ or to the second trunk group if $r_1 > r_0$. The probability of being routed incorrectly then becomes r_1 and r_0 respectively. We assign the link to either group if $r_0 = r_1$.

We note, for the two trunk group example above, that the actual partitioning of the output links is not directly related to the addresses. For example, it is possible to have an output link assigned to trunk group $111\cdots11$ although its address has a smaller Hamming distance from $000\cdots00$ than from $111\cdots11$. The Hamming distance between the codeword and the output link address that the packet exits from represents the number of times that a packet is incorrectly deflected.

We also observe that the partitioning process is sensitive to changes in the input probability; any variation in the input probability requires that the trunk group assignment be repeated. Even if the two trunk groups have the same probability, assignment of output links changes according to the utilization of the SDF. It is also possible to have no output link assigned to a certain packet type.

In the remainder of this paper, we present a scheme that allows the output assignment be made without explicitly calculating the probability of different packet types. We show that by using a generalized addressing scheme, assignment of 2^k trunk groups can be made using only the last k bits of the address. We allow k to be any positive integer such that $k \leq n$. As the number of trunk groups changes, the output links can be assigned with a minimal amount of computation. If the uniform input probability is maintained, the assignment is independent of the offered load.

Next, we illustrate the basic idea of this paper using two packet types with equal probability.

3 Assignment of Two Trunk Groups

In this section, we show that with some hardware modifications, the probability of error in the SDF can be improved for two packet types of equal probability. At the same time, trunk group assignment can be made based only on the last bit of the output address, and this assignment is independent of the offered load. We refer to this switch as the *Modified SDF (MSDF).*

Assume that two packet types with the code words $000\cdots00$ and $111\cdots11$ are given. Assume also that the probability of these two packet types is equal and is uniform over all input links. From the connection pattern in the Omega network, we conclude that the type probabilities of the two input links is equal for S_{lm} for any l and m in the SDF. If one input link of S_{lm} has type probability (p, q), the other input link must also have type probability (p, q). This can be seen from the fact that the paths of the two inputs to S_{lm} can be traced back in an identical manner to the switching elements at stage 1 where the input probability of the two packet types are equal by assumption.

Now, consider S_{0n}. Assume that the two input links to S_{0n} have type probabilities (p, q). Then the two input links of $S_{N/2-1,n}$ have type probabilities (q, p). In fact, we can make a general statement that for every k, if S_{kn} has the type probability (p, q), then $S_{N/2-1-k,n}$ has the type probability (q, p). This can be seen as follows: Given the codeword $000\cdots00$ and $111\cdots11$, for any l and k, the of paths terminating at S_{kn} is mirror-image symmetric to the tree of paths terminating at $S_{N/2-1-k,n}$ and all the switches at stage 1 have the same type probability on their input links.

Given a code word c, a path between S_{l1} and S_{kn} is determined by a binary sequence $b = b_1 b_2 ... b_{n-1}$. In describing the path, b_m is 1 if the preference at stage m is not met, 0 otherwise. b contains the sequence of routing errors made in the path and hence is termed as the *syndrome* of the path. We note that for a fixed k, the paths between S_{l1} and S_{kn} have the same syndrome with respect to $000 \cdots 00$ (or $111 \cdots 11$) among all l's such that $0 \leq l \leq N/2-1$, and this is the same as the syndrome for the path between $S_{N/2-1-l,1}$ and $S_{N/2-1-k,n}$ with respect to $111 \cdots 11$ (or $000 \cdots 00$). We note that k can be written as $b_1 b_2 .... b_{n-1}$ and $N/2 - 1 - k = (2^{n-1} - 1) \oplus b_1 b_2 .... b_{n-1}$ where $\oplus$ is binary addition performed bit by bit.

Assume that the two inputs to S_{kn} have type probabilities (p, q). Without loss of generality, assume that $p > q$. The probability of error for S_{kn} is q if $p^2 \geq 2q - q^2$ or $\frac{1}{2}(p^2 + q^2)$ if $p^2 < 2q - q^2$, as shown in Figure 2. From the figure, we note that the assignment of output links is dependent on the values of p and q. The probability of error for $S_{N/2-1-k,n}$ is the same.

Assume next that the SDF is modified such that one input to S_{kn} is exchanged with one input to $S_{N/2-1-k,n}$ as shown in Figure 3. Then, the probability of error for each switching element becomes pq regardless of p and q. This is seen by the fact that $p + q - pq \geq pq$ for all $0 \leq p, q \leq 1$. This probability is clearly less than or equal to that of the previous case since $q \geq pq$ and $\frac{1}{2}(p^2 + q^2) \geq pq$.

This suggests that if we modify the connection patterns for S_{kn} and $S_{(N/2-1-k),n}$ for $0 \leq k \leq N/2 - 1$, the probability of error is improved. We call the resulting switch the *Modified SDF (MSDF)*. Another important feature of MSDF is that assigning an output link to a trunk group can be done based on the last bit of the address, regardless of the input probability of the switching element that the link is connected to. This assignment remains the same for all offered load values.

This example illustrates that a certain combination of input probabilities at the last stage results in a better probability of error. Furthermore, this combination allows the assignment of trunk groups to be made based entirely on the last bit of the output address. We note, however, that the MSDF requires changes in the connection pattern of the switch, and these changes would depend on the number of packet types that need to be supported. As a result, different MSDF must be used for a different number of packet types, and this is clearly not a desirable property. Therefore, we develop a new link addressing schemes which allows for improvement in the probability of error without hardrare changes.

In Section 4, we expand this idea to $K = 2^k$ trunk groups for $1 \leq K \leq N$. A new link addressing scheme is formulated such that the output assignment can be made based on the last k bits of the address without requiring hardware changes. In Section 4, we employ the Banyan network, which is topologically equivalent to the Omega network.

4 Generalized Trunk Group Assignment

In this section, we generalize the number of trunk groups to $K = 2^k$ where $k \leq n$. For K trunk groups, let $c^0, c^2, \cdots, c^{K-1}$ be the respective code words. Define a function $pref(\cdot, \cdot)$ such that

$$pref(c^i, m) = c^i_m$$

where c^i_m is the m^{th} bit in c^i and where $pref(c^i, m)$ corresponds to the preference indicated by c^i at stage m. Define the sets $R(0, m)$ and $R(1, m)$ such that

$$\begin{aligned} R(0, m) &= \{ i \mid pref(c^i, m) = 0 \} \\ R(1, m) &= \{ i \mid pref(c^i, m) = 1 \}. \end{aligned}$$

Assume that all c^i's $\in R(0, m)$ have the same priority for each m. The same is true for $R(1, m)$. Let $(p^0_{2l,m}, p^1_{2l,m}, \cdots, p^{K-1}_{2l,m})$ be the type probability at input 0 of S_{lm}, and $(p^0_{2l+1,m}, p^1_{2l+1,m}, \cdots, p^{K-1}_{2l+1,m})$ the type probability at input 1. Similarly, we define $(q^0_{2l,m}, q^1_{2l,m}, \cdots, q^{K-1}_{2l,m})$

to be the type probability at output 0 of S_{lm}, and $(q^0_{2l+1,m}, q^1_{2l+1,m}, \cdots, q^{K-1}_{2l+1,m})$ to be the type probability at output 1. Then, for $0 \leq i \leq K-1$, $1 \leq m \leq n$, and $0 \leq l \leq N/2-1$,

$$q^i_{2l,m} = \begin{cases} \text{if } pref(c^i, m) = 0, \\ p^i_{2l,m} + p^i_{2l+1,m} - \frac{1}{2}\, p^i_{2l,m} \sum\limits_{j \in R(0,m)} p^j_{2l+1,m} - \frac{1}{2} p^i_{2l+1,m} \sum\limits_{j \in R(0,m)} p^j_{2j,m}; \\ \\ \text{if } pref(c^i, m) = 1, \\ \frac{1}{2} p^i_{2l,m} \sum\limits_{j \in R(1,m)} p^j_{2l+1,m} + \frac{1}{2} p^i_{2l+1,m} \sum\limits_{j \in R(1,m)} p^j_{2l,m} \end{cases}$$

$$q^i_{2l+1,m} = \begin{cases} \text{if } pref(c^i, m) = 0, \\ \frac{1}{2} p^i_{2l,m} \sum\limits_{j \in R(0,m)} p^j_{2l+1,m} + \frac{1}{2} p^i_{2l+1,m} \sum\limits_{j \in R(0,m)} p^j_{2l,m}; \\ \\ \text{if } pref(c^i, m) = 1, \\ p^i_{2l,m} + p^i_{2l+1,m} - \frac{1}{2} p^i_{2l,m} \sum\limits_{j \in R(1,m)} p^j_{2l+1,m} - \frac{1}{2} p^i_{2l+1,m} \sum\limits_{j \in R(1,m)} p^j_{2l,m}. \end{cases} \tag{1}$$

For the MSDF with K trunk groups, it is necessary to change the connection patterns for the last k stages. This implies that different configurations of MSDF must be used for different number of trunk groups, which is clearly not desirable. Next, we attempt to develop a self-routing, multichannel switch that does not involve hardware changes as the number of trunk groups varies. While it is topologically equivalent to the Omega network used in [5] and [6], we found it more convenient to use another form of Banyan network in our development. Detailed description of this Banyan network can be found in [8]. An example of this Banyan Network is shown in Figure 4.

To gain facility with what follows, readers should refer to Figure 5. Consider a Banyan network with $N = 2^n$ input links, and hence n stages. Given $K = 2^k$ trunk groups, we note that the first $n-k$ stages of the Banyan network consist of K disjoint Banyan networks, each with 2^{n-k} input lines. We label the $n-k$ staged Banyan networks as $B_{1,j}$ starting from the top where $0 \leq j \leq K-1$. An example of this with $K = 2$ is shown in Figure 5. We term the index j as the *level* of the Banyan network.

Define H_l for $0 \leq l \leq K-1$ to be the set of the l^{th} output links from each $B_{1,j}$. If all switching elements between stages $n-k+1$ and n whose inputs are driven by H_l are isolated, they would themselves form a separate Banyan network with K inputs and k stages. We term this collection as $B_{2,l}$ for $0 \leq l \leq 2^{n-k}-1$. In $B_{2,l}$, input link j is output link l from $B_{1,j}$.

Next, we focus on the first $n-k$ stages of the Banyan network. Let $c^0, c^1, ..., c^{K-1}$ be the set of K code words that are $n-k$ bits long. We assign for each $B_{1,j}$, a different ordering of the code words such that a packet destined for the i^{th} trunk group is given $c^{i \oplus j}$ if it enters at level j. Such assignments of code words are shown in Table 1. As a packet enters the Banyan network, a code word is assigned according to the table. Then, the packet traverses through the $n-k$ stages of $B_{1,j}$ according to the preference specified in the code word. **Theorem 1** Given 2^k trunk groups with equal probability, the code word assignment in Table 1 results in,

$$\left(q^0_{l,n-k}(s), q^1_{l,n-k}(s), \cdots, q^{K-1}_{l,n-k}(s)\right) = \left(q^{0 \oplus j}_{l,n-k}(s \oplus j), q^{1 \oplus j}_{l,n-k}(s \oplus j), \cdots, q^{(K-1) \oplus j}_{l,n-k}(s \oplus j)\right)$$

where $q^i_{l,n-k}(s)$ is the probability of packets belonging to the i^{th} trunk group observed at output link l, stage $n-k$,

Proof Given $\left(q^0_{l,n-k}(0), q^1_{l,n-k}(0), \cdots, q^{K-1}_{l,n-k}(0)\right)$, we note that $q^{i \oplus j}_{l,n-k}(j) = q^i_{l,n-k}(0)$. This can be seen by the fact that trunk group i at level 0 and trunk group $i \oplus j$ at level j have the same code word $c^i (= c^{i \oplus j \oplus j})$, that every level is topologically identical, and that the input probability is the same among all trunk groups. Since this is true for every i and j, the theorem follows. $\square$

Assume that for output link l at level 0, we have $q^i_{l,n-k}(0) = p^i$ for $0 \leq i \leq K-1$. Then, based on Theorem 1, we can determine $q^i_{l,n-k}(s)$ for $0 \leq s \leq K-1$ as shown in Table 2. Hence, Table 2 shows the type probability for H_l, the collection of inputs to $B_{2,l}$.

Let us now focus on the remaining k stages of the Banyan network. Consider $B_{2,l}$. We determine the code word for trunk group i in the following manner: If the binary representation of i is $i_{k-1}i_{k-2}\cdots i_1 i_0$ where i_0 is the least significant bit, the code word for trunk group i becomes $i_0 i_1 \cdots i_{k-2} i_{k-1}$; the preference of trunk group i at stage m of $B_{2,l}$ is i_{m-1}. Recall that this is a different code word from c^i which is used for the first $n-k$ stages. Thus,

$$\begin{aligned} R(0,\lambda) &= \{\, i \mid pref(i,\lambda) = 0\} \\ R(1,\lambda) &= \{\, i \mid pref(i,\lambda) = 1\}. \end{aligned} \tag{2}$$

We label the input links to $B_{2,l}$ between 0 and $K-1$ starting from the top. We use the following convention to label the remaining links in $B_{2,l}$: If a switching element has its upper input link labeled as a and the lower link labeled as b, then the upper output links of the switching element is also labeled as a and the lower output link labeled as b. An example of this is also shown in Figure 5.

Let $r^i_{h,\lambda}$ be the probability of trunk group i on output link h at stage λ of $B_{2,l}$. By definition, we set $r^i_{h,0}$ to be the probability at input link h for trunk group i. Then from Table 2, $r^i_{0,0} = p^i$ and $r^i_{j,0} = p^{i\oplus j}$.

Theorem 2 For $B_{2,l}$ with the link labels defined as before, let $r^{i\oplus j}_{\lambda} \triangleq r^{i\oplus j}_{0,\lambda}$ where $1 \leq \lambda \leq k$ and $0 \leq i,j \leq K-1$. Then, $r^i_{j,\lambda} = r^{i\oplus j}_{\lambda}$.

Proof We note first that if one input to a switching element at stage λ is labeled as j, then the other input is labeled as $j \oplus 2^{\lambda-1}$. At stage λ, every packet belonging to the trunk groups whose labels are in the form of i such that $i \pmod{2^\lambda} < 2^{\lambda-1}$ has the preference zero since the λ^{th} bit in i is zero. Otherwise, the packet has the preference one. To prove the theorem, we use the following induction.

For $\lambda = 1$, two inputs (and also outputs) of a switching element have the labels in the form of $2g$ and $2g+1$ for $0 \leq g \leq K/2-1$. Note that

$$\begin{aligned} R(0,1) &= \{\, i \mid i \text{ in the form of } 2h\} \\ R(1,1) &= \{\, i \mid i \text{ in the form of } 2h+1\}. \end{aligned} \tag{3}$$

Based on (1), for output link 0 at stage 1, we have

$$\begin{aligned} r^{2h}_{0,1} &= r^{2h}_{0,0} + r^{2h}_{1,0} - \tfrac{1}{2}\, r^{2h}_{0,0} \sum_{m\in R(0,1)} r^m_{1,0} - \tfrac{1}{2}\, r^{2h}_{1,0} \sum_{m\in R(0,1)} r^m_{0,0} \\ &= r^{2h}_{0,0} + r^{2h}_{1,0} - \tfrac{1}{2}\, r^{2h}_{0,0} \sum_m r^{2m}_{1,0} - \tfrac{1}{2}\, r^{2h}_{1,0} \sum_m r^{2m}_{0,0} \\ &= p^{2h} + p^{2h+1} - \tfrac{1}{2}\, p^{2h} \sum_m p^{2m+1} - \tfrac{1}{2}\, p^{2h+1} \sum_m p^{2m} \\ r^{2h+1}_{0,1} &= \tfrac{1}{2}\, r^{2h+1}_{0,0} \sum_{m\in R(1,1)} r^m_{1,0} + \tfrac{1}{2}\, r^{2h+1}_{1,0} \sum_{m\in R(1,1)} r^m_{0,0} \\ &= \tfrac{1}{2}\, r^{2h+1}_{0,0} \sum_m r^{2m+1}_{1,0} + \tfrac{1}{2}\, r^{2h+1}_{1,0} \sum_m r^{2m+1}_{0,0} \\ &= \tfrac{1}{2}\, p^{2h+1} \sum_m p^{2m} + \tfrac{1}{2}\, p^{2h} \sum_m p^{2m+1}. \end{aligned} \tag{4}$$

where h is any integer such that $0 \leq h \leq K/2-1$.

Similarly, we repeat the above calculations for every link j at stage 1. Then,

$$r^i_{j,1} = \begin{cases} \text{if } i+j \text{ is even,} \\ r^i_{j,0} + r^i_{j\oplus 1,0} - \frac{1}{2}\, r^i_{j,0} \displaystyle\sum_{m\in R(pref(i),1)} r^m_{j\oplus 1,0} - \frac{1}{2}\, r^i_{j\oplus 1,0} \sum_{m\in R(pref(i),1)} r^m_{j,0}; \\ \\ \text{if } i+j \text{ is odd,} \\ \frac{1}{2}\, r^i_{j,0} \displaystyle\sum_{m\in R(pref(i),1)} r^m_{j\oplus 1,0} + \frac{1}{2}\, r^i_{j\oplus 1,0} \sum_{m\in R(pref(i),1)} r^m_{j,0}. \end{cases} \tag{5}$$

In (5), we have used the fact that when the preference for a packet of type i is satisfied, it is routed to output link j where $i+j$ is even. On the other hand, $i+j$ is odd if the preference is not satisfied. For example, if i is an even number, its preference is 0. In order to route a packet belonging to trunk group i correctly, the packet is routed to output 0 which has label j that is an even number. Hence, the correct routing will result in $i+j$ as an even number.

Using the fact that $2i+1 = 2i \oplus 1$, that $2i+1 \oplus 1 = 2i$, and that $r^i_{j,0} = p^{i\oplus j}$, (5) now becomes,

$$r^i_{j,1} = \begin{cases} \text{if } i+j \text{ is even} \\ p^{i\oplus j} + p^{i\oplus j\oplus 1} - \frac{1}{2}\, p^{i\oplus j} \displaystyle\sum_m p^{2m\oplus 1} - \frac{1}{2}\, p^{i\oplus j\oplus 1} \sum_m p^{2m}; \\ \\ \text{if } i+j \text{ is odd,} \\ \frac{1}{2}\, p^{i\oplus j} \displaystyle\sum_m p^{2m\oplus 1} + \frac{1}{2}\, p^{i\oplus j\oplus 1} \sum_m p^{2m}. \end{cases} \tag{6}$$

From (4) and (6), we can conclude that $r^i_{j,1} = r^{i\oplus j}_{0,1}$ and this proves the theorem with $\lambda = 1$.

We want to show next that for $2 \le \lambda \le k$, if the theorem is true for $\lambda - 1$, it is also true for λ. Assume that the theorem is true for $\lambda - 1$.

Let $\beta \triangleq 2^{\lambda-1}$. Then for $0 \le g < 2^{k-\lambda}$ and $0 \le h < \beta$, we have,

$$\begin{aligned} pref(2^\lambda g + h, \lambda) &= 0 \\ pref((2^\lambda g + h) \oplus \beta, \lambda) &= 1. \end{aligned} \tag{7}$$

Recall that the two inputs to any switching element have labels in the form of j and $j \oplus \beta$. Based on this fact, we can write the output probabilities of trunk groups $2^\lambda g + h$ and $2^\lambda g + h + \beta$ for link 0 at stage λ as,

$$\begin{aligned} r^{2^\lambda g+h}_\lambda &= r^{2^\lambda g+h}_{0,\lambda} \\ &= r^{2^\lambda g+h}_{0,\lambda-1} + r^{2^\lambda g+h}_{\beta,\lambda-1} - \frac{1}{2}\, r^{2^\lambda g+h}_{0,\lambda-1} \sum_{m\in R(pref(2^\lambda g+h),\lambda)} r^m_{\beta,\lambda-1} \\ &\quad - \frac{1}{2}\, r^{2^\lambda g+h}_{\beta,\lambda-1} \sum_{m\in R(pref(2^\lambda g+h),\lambda)} r^m_{0,\lambda-1} \\ &= r^{2^\lambda g+h}_{\lambda-1} + r^{2^\lambda g+h\oplus\beta}_{\lambda-1} - \frac{1}{2}\, r^{2^\lambda g+h}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} \\ &\quad - \frac{1}{2}\, r^{2^\lambda g+h\oplus\beta}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1} \\ r^{2^\lambda g+h+\beta}_\lambda &= r^{2^\lambda g+h+\beta}_{0,\lambda} \\ &= \frac{1}{2}\, r^{2^\lambda g+h+\beta}_{0,\lambda-1} \sum_{m\in R(pref(2^\lambda g+h\oplus\beta),\lambda)} r^m_{\beta,\lambda-1} + \frac{1}{2}\, r^{2^\lambda g+h+\beta}_{\beta,\lambda-1} \sum_{m\in R(pref(2^\lambda g+h\oplus\beta),\lambda)} r^m_{0,\lambda-1} \\ &= \frac{1}{2}\, r^{2^\lambda g+h+\beta}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1} + \frac{1}{2}\, r^{2^\lambda g+h}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} \end{aligned} \tag{8}$$

where we have used the facts that

$$\begin{aligned}(2^\lambda g + h) \oplus \beta &= 2^\lambda g + h + \beta \\ (2^\lambda g + h) \oplus \beta \oplus \beta &= 2^\lambda g + h.\end{aligned}$$

Note next the fact that when the preference for a packet of type i is satisfied, it is routed to output link j where $(i \oplus j) \bmod 2^\lambda < \beta$. $(i \oplus j) \bmod 2^\lambda \geq \beta$ if the preference is not satisfied. Then, for every link i and stage j,

$$r^i_{j,\lambda} = \begin{cases} \text{if } pref(i) \text{ is satisfied,} \\ r^i_{j,\lambda-1} + r^i_{j\oplus\beta,\lambda-1} - \frac{1}{2}\, r^i_{j,\lambda-1} \displaystyle\sum_{m\in R(pref(i),\lambda)} r^m_{j\oplus\beta,\lambda-1} - \frac{1}{2}\, r^i_{j\oplus\beta,\lambda-1} \displaystyle\sum_{m\in R(pref(i),\lambda)} r^m_{j,\lambda-1}; \\ \text{if } pref(i) \text{ is not satisfied,} \\ \frac{1}{2}\, r^i_{j,\lambda-1} \displaystyle\sum_{m\in R(pref(i),\lambda)} r^m_{j\oplus\beta,\lambda-1} + \frac{1}{2}\, r^i_{j\oplus\beta,\lambda-1} \displaystyle\sum_{m\in R(pref(i),\lambda)} r^m_{j,\lambda-1}, \end{cases} \tag{9}$$

which can be rewritten as,

$$r^i_{j,\lambda} = \begin{cases} \text{if } pref(i) \text{ is satisfied,} \\ r^{i\oplus j}_{\lambda-1} + r^{i\oplus j\oplus\beta}_{\lambda-1} - \frac{1}{2}\, r^{i\oplus j}_{\lambda-1} \displaystyle\sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} - \frac{1}{2}\, r^{i\oplus j\oplus\beta}_{\lambda-1} \displaystyle\sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1}; \\ \text{if } pref(i) \text{ is not satisfied,} \\ \frac{1}{2}\, r^{i\oplus j}_{\lambda-1} \displaystyle\sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} + \frac{1}{2}\, r^{i\oplus j\oplus\beta}_{\lambda-1} \displaystyle\sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1}. \end{cases} \tag{10}$$

Hence, from (8) and (10), we conclude that $r^i_{j,\lambda} = r^{i\oplus j}_{0,\lambda} = r^{i\oplus j}_\lambda$ and the theorem follows. □

Theorem 3 Let the link labels for $B_{2,l}$ and r^i_λ be defined as in Theorem 2. Then, for $1 \leq \lambda \leq k$, $0 \leq i < 2^{k-\lambda}$, and $0 \leq j < 2^\lambda$, $r^{2^\lambda i}_\lambda \geq r^{2^\lambda i+j}_\lambda$.

Proof Let $0 \leq g < 2^{k-\lambda}$, $\beta = 2^{\lambda-1}$, and $0 \leq h < \beta$. Again we use induction to prove this theorem. Let $\lambda = 1$. From (4), we have $r^{2g}_{0,1} \geq r^{2g+1}_{0,1}$. Then using Theorem 2, $r^{2g}_1 \geq r^{2g+1}_1$.

Assume that the theorem is true for $\lambda - 1$. Then, using (8), we can see that

$$r^{2^\lambda g+h}_\lambda \geq r^{2^\lambda g+h+\beta}_\lambda. \tag{11}$$

Furthermore,

$$\begin{aligned} & r^{2^\lambda g}_\lambda - r^{2^\lambda g+h}_\lambda \\ = \;& r^{2^\lambda g}_{\lambda-1} + r^{2^\lambda g\oplus\beta}_{\lambda-1} - \frac{1}{2}\, r^{2^\lambda g}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} - \frac{1}{2}\, r^{2^\lambda g\oplus\beta}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1} \\ & - r^{2^\lambda g+h}_{\lambda-1} - r^{2^\lambda g+h\oplus\beta}_{\lambda-1} + \frac{1}{2}\, r^{2^\lambda g+h}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1} \\ & + \frac{1}{2}\, r^{2^\lambda g+h\oplus\beta}_{\lambda-1} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1} \\ = \;& (r^{2^\lambda g}_{\lambda-1} - r^{2^\lambda g+h}_{\lambda-1})(1 - \frac{1}{2} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w\oplus\beta}_{\lambda-1}) \\ & + (r^{2^\lambda g\oplus\beta}_{\lambda-1} - r^{2^\lambda g+h\oplus\beta}_{\lambda-1})(1 - \frac{1}{2} \sum_{v=0}^{2^{k-\lambda}-1} \sum_{w=0}^{2^\lambda-1} r^{2^\lambda v+w}_{\lambda-1}) \geq 0 \end{aligned} \tag{12}$$

where the inequality holds by the induction hypothesis.

We note that i and g have the same range. Furthermore, since we can see that any j such that $0 \leq j < 2^{\lambda}$ can be expressed in the form of h or $h + \beta$, the theorem follows. □

Finally, let $\lambda = k$, $i = 0$, and $0 \leq j < 2^k$. Then Theorem 3 states that for link 0, we have, $r_k^0 \geq r_k^j$, which implies $r_{0,k}^0 \geq r_{0,k}^j$. Hence, link 0 should be assigned to trunk group 0. Noting that, for every link j,

$$r_{j,k}^i = r_{0,k}^{i \oplus j} = r_k^{i \oplus j},$$

we conclude that $r_{j,k}^j = r_k^0$ is the largest probability for this link. So in order to minimize the probability of error under this scheme, we assign output link j of $B_{2,l}$ to trunk group j for every l. Thus the assignment of trunk groups can be done based on the output addresses of the links. Since there are $K = 2^k$ output links in $B_{2,l}$, the last k bits of the output address is uniquely identify each link.

5 Conclusions

In this paper, a generalized addressing scheme is proposed for assigning arbitrary 2^k trunk groups in multichannel switches. Assuming that all trunk groups have equal probability, an output link can be assigned to a trunk group using only the last k bits of address in order to minimize the error probability. We note that this assignment is independent of the offered load.

References

[1] Jaffe, J.M., Moss, F.H., and Weingarten, R.A., "NA Routing: Past, Present, and Possible Future," *IBM Sys. J.,*, Volume 22, No. 4, Dec, 1983, pp. 417-434.

[2] Liew, S., and Lu, K., "Performance Analysis of Asymmetric Packet Switch Modules with Channel Grouping," *Proceedings of IEEE INFOCOM '90*, June, 1990, pp. 668-676.

[3] Lin, A., and Silvester, J.A., "Fixed-Node Routing and Its Performance in ATM Networks," *Proceedings of IEEE INFOCOM '90*, June, 1990, pp. 803-810.

[4] Pattavina, A., "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE JSAC*, Volume 6, No. 9, Dec, 1988, pp. 1489-1499.

[5] Cruz, R.L., "The Statistical Data Fork: A Class of Broadband Multichannel Switches," *IEEE Transactions on Communications*, to appear.

[6] Bellur, B., and Sasaki, G.H., "Buffered Statistical Data Fork," *Technical Report C-90-2*, The University of Texas at Austin, Oct, 1990. 1372-1380

[7] Bertsekas, D., and Gallager, R., *Data Networks*, Prentice Hall, Inc. New Jersey, 1987.

[8] Hui, J.Y., *Switching and Traffic Theory for Integrated Broadband Networks,* Kluwer Academic Publishers, 1990.

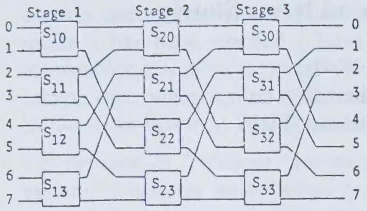
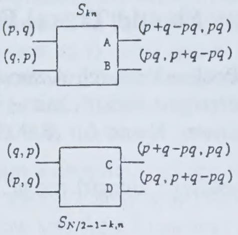
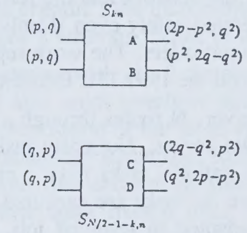

Stage 1
Stage 2
Stage 3
S_{10}
S_{11}
S_{12}
S_{13}
S_{20}
S_{21}
S_{22}
S_{23}
S_{30}
S_{31}
S_{32}
S_{33}
0
1
2
3
4
5
6
7

Figure 1: An Example of the SDF with $N = 8$


$S_{k,n}$
(p, q)
(q, p)
A
B
$(p+q-pq, pq)$
$(pq, p+q-pq)$
(q, p)
(p, q)
C
D
$(p+q-pq, pq)$
$(pq, p+q-pq)$
$S_{N/2-1-k,n}$

Link	Trunk Group	P_{error}
A	Group 1	pq
B	Group 2	pq
C	Group 1	pq
D	Group 2	pq

Figure 3: Trunk Group Assignment in MSDF


$S_{k,n}$
(p, q)
(p, q)
A
B
$(2p-p^2, q^2)$
$(p^2, 2q-q^2)$
(q, p)
(q, p)
C
D
$(2q-q^2, p^2)$
$(q^2, 2p-p^2)$
$S_{N/2-1-k,n}$

	$p^2 \geq 2q-q^2$		$p^2 < 2q-q^2$	
Link	Trunk Group	P_{error}	Trunk Group	P_{Error}
A	Group 1	q^2	Group 1	q^2
B	Group 1	$2q-q^2$	Group 2	p^2
C	Group 2	$2q-q^2$	Group 1	p^2
D	Group 2	q^2	Group 2	q^2

Figure 2: Trunk Group Assignment in the SDF

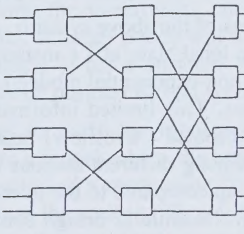


Figure 4: An Example of Banyan Network with $N = 8$

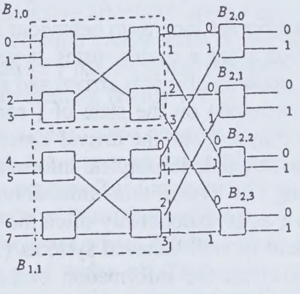

$B_{1,0}$
$B_{1,1}$
$B_{2,0}$
$B_{2,1}$
$B_{2,2}$
$B_{2,3}$
0
1
2
3
4
5
6
7

Figure 5: Partitioned Banyan Network with $K = 2$

Table 1: Code Words

level	Trunk Group						
	0	$\cdots$	i	$\cdots$	j	$\cdots$	$K-1$
0	c^0	$\cdots$	c^i	$\cdots$	c^j	$\cdots$	c^{K-1}
1	c^1	$\cdots$	$c^{i\oplus 1}$	$\cdots$	$c^{j\oplus 1}$	$\cdots$	$c^{K-1\oplus 1}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	c^i	$\cdots$	c^0	$\cdots$	$c^{j\oplus i}$	$\cdots$	$c^{K-1\oplus i}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
j	c^j	$\cdots$	$c^{i\oplus j}$	$\cdots$	c^0	$\cdots$	$c^{K-1\oplus j}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$K-1$	c^{K-1}	$\cdots$	$c^{i\oplus K-1}$	$\cdots$	$c^{j\oplus K-1}$	$\cdots$	c^0

Table 2: Output Probabilities at link l

level	Trunk Group						
	0	$\cdots$	i	$\cdots$	j	$\cdots$	$K-1$
0	p^0	$\cdots$	p^i	$\cdots$	p^j	$\cdots$	p^{K-1}
1	p^1	$\cdots$	$p^{i\oplus 1}$	$\cdots$	$p^{j\oplus 1}$	$\cdots$	$p^{(K-1)\oplus 1}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	p^i	$\cdots$	p^0	$\cdots$	$p^{j\oplus i}$	$\cdots$	$p^{(K-1)\oplus i}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
j	p^j	$\cdots$	$p^{i\oplus j}$	$\cdots$	p^0	$\cdots$	$p^{(K-1)\oplus j}$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
$K-1$	p^{K-1}	$\cdots$	$p^{i\oplus(K-1)}$	$\cdots$	$p^{j\oplus(K-1)}$	$\cdots$	p^0

Decentralized Control of Networks: Bad News Should Travel Fast, Good News Slowly

René K. Boel - Research Associate N.F.W.O.
Lab. Theoretische Elektriciteit - Universiteit Gent
Grote Steenweg Noord 6 B-9052 Zwijnaarde-GENT
Belgium
e-mail: boel@gpx.autoctrl.rug.ac.be

Abstract

This contribution treats communication network protocols as decentralized control laws for a (possibly very large) Markovian model. Alternatively the models considered describe road traffic networks or manufacturing systems, where the size of the network and the reliability requirements impose a constraint that the system can be operated safely even if only local state information is available to each (or some) of the local controllers. The work reported here is a sequel to work reported at the 1986 and 1991 CDC and the 1991 ITC Conferences.

Consider a system with N stations (N LAN's with a local server, N routes through a city). Each station has its own arrival stream of tasks and its own server. Normally tasks are served on the local server. However in case of overload at station i tasks from i may be rerouted to some other server $j \in S_i$, the rerouting set. There the tasks are executed more slowly (due to extra transmission delay, server not well adapted to rerouted job, etc.). Hence rerouting decisions should be taken carefully: rerouting too often destabilises the overall system by spreading the overload to otherwise not overloaded stations. This destabilisation is typical for decentralized systems (cf. ALOHA). It manifests itself as rare, large excursions, described as oscillatory behaviour in a number of experimental papers.

In order to analyze the long term behaviour and the robustness of the above systems, assume that each station i has a router, using measurements of the local state at i (instantaneous queue length, possibly also past arrival and service times for non-exponential models) as well as limited information on the state of (some other) stations. This limited information on other stations may e.g. be the arrival times at station i of rerouted (overflow) tasks from station j, or occasional, incomplete information exchanges among different stations in each others rerouting set. Even if this information exchange is cheap compared to the overall cost of the system (as is undoubtedly often the case) it is still worthwhile to design controllers so that they lead to well behaved systems (stable) even if there would e.g. be temporarily large error rates on the information exchange between stations. No efficient method is known, except under special assumptions, for designing an optimal decentralized controller.

Consider first the case of $N=2$ stations. Assume Poisson arrivals (rate λ_i) and exponential service times with parameters $\mu_{i,j}$ for tasks of station i on server j. Typically $\mu_{ij} < \mu_{ii}$ for $j \neq i$. To obtain a simple Markovian model with state space $(Q_{1,t}, Q_{2,t})$ assume that rerouted tasks from i to j generate (in j) μ_{jj} / μ_{ij} tasks. If $\lambda_i < \mu_{i,i}$, but $\lambda_i \geq \mu_{i,j}$ for $i \neq j$, and if the decentralized controllers use a threshold policy rerouting to j whenever the local queue exceeds the threshold B_i, then large deviations theory (discontinuities in the drift terms are allowed using the results of Dupuis, Ishii and Soner) shows that this non-ergodic system will escape from a neighbourhood of the origin in the scaled state space $(Q_{1,nt}/n, Q_{2,nt}/n)$ via the edges: first $Q_{i,t}$ becomes large while $Q_{j,t}$ $(j \neq i)$ remains small; then as a result of rerouting from i to j $Q_{j,t}$ grows large very quickly; large deviations theory shows that this second build-up phase occurs at an exponentially faster rate than the first phase of the escape. To stabilize the system, without transmitting global information too often, it is necessary to react very quickly during this second phase of the escape. The bad news that while the threshold of queue i has been reached (local information at i), also queue j has started growing (information available at j) has to be transmitted very fast, namely at a rate determined by the escape rate when station i has reached threshold. This rate (itself still exponentially small in the scale factor n) is exponentially larger than the escape rate from near the origin.

For the case of 2 stations, it is easy to design a decentralized controller which stabilizes the system. Whenever an overflow task from station i arrives at j the rerouting from j to i is stopped for a duration T_1 ; if another overflow task from i reaches j before the end of the no-rerouting interval then the no-rerouting interval is extended by T_2, etc (cf. exponential backoff in Ethernet). This controlled queueing system can be modelled as a Markov process with queue lengths and remaining no-rerouting interval as states. Drift analysis as in Hajek's analysis of adaptively stabilized ALOHA shows that proper choice of the T_i's makes the system ergodic (the increments of the no-rerouting interval can be kept bounded as required in Hajek's analysis thanks to the essentially linear behaviour of overloaded queues).

For systems with $N > 2$ the above policy at least requires that if i sends overflow tasks to j, then j also sends overflow tasks to i. On the other hand taking $S_i = \{j\}$ and $S_j = \{i\}$ leads to a decomposition of the large system into decoupled 2-station systems, and loses most of the load balancing advantage. Rerouting sets S_i have to be carefully selected. Signalling has to be taken into account as well as load balancing. Intuitively one expects that sequencing overflow tasks from i over S_i according to a round robin policy will give better load balancing and better signalling performance than random choice. It is an open problem whether the optimal round robin sequences for load balancing are also optimal for signalling.

Large deviations analysis of the Markov model even for $N=2$ with no-rerouting intervals becomes intractable. Intuition however suggests that the behaviour will still be very oscillatory: the no-rerouting intervals only become active after both queues have reached their threshold. A better policy will involve active information transmission on the status of station i, to station j, for a duration T, as soon as station i has received overflow tasks from j. Analysis of policies of this type has not led to interesting results yet.

In order to study the robustness of the system against slow changes in the arrival rates, consider a model where the arrival processes are Markov modulated Poisson (arrival rates are function of a modulating finite-state Markov process Z_t). Large deviations analysis now should use the scaling ($Q_{1,nt}/n$, $Q_{2,nt}/n$, Z_{nt}). Results of Bezuidenhout show that, as intuitively expected, the most likely escape path will be trajectories where Z_t takes values corresponding to high arrival rates for a long time duration (of order n). The escape rates from the neighbourhood of the good operating point (near the origin) are essentially determined by the escape rates for the "worst" value of Z. The long term behaviour of the controlled system will be very sensitive to the knowledge of the modulating state Z_t. Station i can efficiently estimate its own arrival rate. Other stations $j \in S_i$ cannot estimate the rate at i efficiently since they only see overflow traffic from i, but whwnever station i is heavily loaded by its own traffic, the no-rerouting intervals used by other stations for sending traffic to i should adaptively be increased significantly (remember that transients of a heavily loaded queue are very slow). This indicates that active information exchange from an overloaded station i to stations in S_i is necessary. It is now clearly important not to send estimates of arrival rates at i that are too optimistic. While estimated increases in λ_i should be transmitted fast, estimated decreases in λ_i should be transmitted slowly. Transmitting a slightly underestimated value of λ_i will cause an exponentially large increase in the (small) rate of escape of the system, or at least (if the system is stabilized) make its behaviour much more oscillatory.

Packet Routing in Networks with Long Wires

Ronald I. Greenberg and H.-C. Oh
Department of Electrical Engineering
University of Maryland
College Park, MD 20742
rig@eng.umd.edu and ohc@eng.umd.edu

September 23, 1992

Abstract

In this paper, we examine the packet routing problem for networks with wires of differing length. We consider this problem in a network independent context, in which routing time is expressed in terms of "congestion" and "dilation" measures for a set of packet paths. We give, for any constant $\epsilon > 0$, a randomized on-line algorithm for routing any set of N packets in $O((C \lg^{\epsilon}(Nd) + D \lg(Nd))/ \lg\lg(Nd))$ time, where C is the maximum congestion and D is the length of the longest path, both taking wire delays into account, and d is the longest path in terms of *number* of wires. We also show that for edge-simple paths, there exists a schedule (which could be found off-line) of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$, where d_{max} is the maximum wire delay in the network. These results improve upon those of Leighton, Maggs, and Rao, which assume that unit time suffices to traverse a wire of any length. Our results also improve upon those of Shmoys, Stein, and Wein for job-shop scheduling as long as we incorporate a technical restriction on the job-shop problem.

1 Introduction

An efficient packet routing algorithm is critical to the design of most large-scale general-purpose parallel computers. One must move data between different locations in an appropriate routing network as quickly as possible and with as little queueing hardware as possible. The packet routing problem has been extensively studied in the past, mostly in the context of specific networks and specific message patterns. Recent works by Leighton, Maggs, Rao, and Ranade have provided very general packet routing results (based on summary measures of the message traffic), which even yield many improvements upon prior analyses of specific networks and message patterns [1, 2, 3]. But these works have made the simplifying assumption that unit time suffices for any transmission of a packet from one network node to another regardless of the actual length of wire connecting the nodes. This assumption becomes less and less tenable as we build larger and larger parallel machines. Hence this paper considers the situation in which an arbitrary delay is associated with each wire.

Except for the introduction of nonunit wire delay, we follow the commonly used *store-and-forward* routing model and the usual graph-based terminology. Packets are atomic objects, which at each time step, either wait in a queue or are in transit on some edge of the network connecting two nodes. Associated with each edge e is an edge delay of $d_e > 0$ time steps required for a packet to traverse that edge, and at any given time, at most one packet can be present on each edge. (There are other interesting routing models, for example allowing the use of transmission lines on which packets can be pipelined, circuit-switching, or wormhole routing, which are not considered in this paper.)

Packets wait in three types of queues. Before routing begins, packets are stored at *initial queues* in the nodes where they are generated. Each time a packet traverses an edge, it enters the *edge queue* at the end of that edge; a packet can begin to traverse an edge only if the queue at the end of that edge is not full. Finally, when a packet reaches its destination, it is placed into a *final queue* at that node. The sizes of the initial and final queues are determined solely by the packet routing problem to be solved, but we seek routing schedules that bound the maximum queue size for edge queues.

We may view the packet routing problem as being comprised of two tasks, selecting a path through the network for each packet and setting a schedule for when packets move and wait. The second task has traditionally been the more difficult one, and it is the focus of this paper. Of course, the selection of paths affects the time and queue size required by a legitimate schedule. For example, the maximum distance d, in number of edges, traveled by any packet is a lower bound on the routing time; this distance is often referred to as the *dilation* in the literature. In fact, the routing time is lower bounded by the maximum over all packet paths of the sum of edge delays along the path. We refer to this measure as the *generalized dilation* D, which differs from d when the unit wire delay assumption is discarded. Similarly, the routing time is lower bounded by the *congestion* c, the maximum over all edges of the number of packets that must traverse the edge over the entire course of the routing, and by the *generalized congestion* C, the maximum over all edges of the number of packets traversing the edge multiplied by the delay of the edge. We also use the notation d_{max} for the maximum over edges e of the edge delay d_e.

Leighton, Maggs, and Rao have given a randomized on-line algorithm for the unit wire delay case, which (with high probability) produces a schedule of length $O(c + d\lg(Nd))$ with queues of size $O(\lg(Nd))$, where N is the number of packets [2]. This naturally implies that in the problem with general edge delays, we could obtain a schedule of length $O(d_{max}(c + d\lg(Nd)))$ by simply using d_{max} time steps to simulate each step of the unit delay algorithm. In Section 2, we give, for any $\epsilon > 0$, an on-line algorithm that produces a schedule of length $O((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd))$ with queues of size $O\left(\frac{\lg(Nd)}{\lg\lg(Nd)}\right)$. This is a significant improvement, since cd_{max} and dd_{max} may be much larger than C and D. It should also be noted that the constants hidden in the O-notation are of modest size at least for $\epsilon = 1$, so the algorithm is practical.

Our on-line algorithm is also an improvement upon the result obtained from the (off-line but polynomial time) algorithm of Shmoys, Stein, and Wein [4] for job-shop scheduling. In job-shop scheduling, the problem input consists of a set of jobs and a set of machines. Each job consists of a sequence of operations, each of which has a specified duration and must be processed on a specified machine. The operations of a job must be processed in order, and each machine can handle at most one operation at a time. We can draw a correspondence between job-shop scheduling and packet routing by thinking of jobs as packets and machines as network edges. The schedule length of Shmoys, Stein, and Wein translated into our notation for packet routing is $O\Big((C+D)\frac{\lg^2(Nd)}{\lg\lg(Nd)}\Big)$. Our superior result for packet routing can be applied to job-shop scheduling as long as we impose the restriction that on any given machine, all operations are of the same duration.

Leighton, Maggs, and Rao have also shown, for unit wire delay, that when the paths traversed by the packets are edge-simple, there exists some schedule of length $O(c+d)$ requiring only constant size queues. This immediately implies existence of a schedule of length $O(d_{max}(c+d))$. Though we have not removed both the $\lg(Nd)$ factor and all dependence on d_{max} in the off-line case, we show in Section 3 that there exists a schedule of length $O\Big((cd_{max}+D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ with queues of size $O(d_{max})$. This result also applies to the restricted form of job-shop scheduling with the additional restriction that no job has more than one operation on a single machine.

2 On-Line Algorithm

Our basic approach to produce a schedule on-line is, as in [2, 4], to first produce an "unconstrained" schedule in which several packets may travel on the same edge at the same time and then "flatten" it into a legitimate schedule. We begin by showing how to produce a schedule of length $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ with queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ when d_{max} is bounded above by a polynomial in N and d; later we refine the result to obtain, for any constant $\epsilon > 0$, a schedule of length $O((C\lg^{\epsilon}(Nd) + D\lg(Nd))/\lg\lg(Nd))$ with queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ and no restriction on d_{max}.

For our initial result, the method for constructing the unconstrained schedule and the analysis of this phase are essentially the same as in the approach of Shmoys, Stein, and Wein. Each packet chooses an integral delay randomly and uniformly from the interval $[1, C]$. A packet that is assigned delay x waits in its initial queue for x time steps and then proceeds to its destination without stopping. Though this may cause more than one packet to traverse a single edge at the same time, it is unlikely that too many will do so:

Lemma 1 (Shmoys, et. al.) *When d_{max} is bounded above by a polynomial in N and d, the strategy of delaying each packet in its initial queue an integral amount chosen randomly and uniformly from $[1, C]$ yields an unconstrained schedule that is of length at most $C+D$ and, with high probability, has no more than $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ packets traversing any edge at any time.*

Proof. We begin by considering the probability p that more than τ packets are present on a particular edge e during a particular time step t. Though packets may spend many time steps traversing e, there are at most C total time units of routing on edge e. Thus, there are at most $\binom{C}{\tau}$ ways to choose τ units of packet routing to occur on edge e at time t. The probability that an individual one of these τ units is scheduled on edge e at time t is at most $1/C$ since each packet chose a delay uniformly at random from C possibilities. If these τ units of routing are all from different packets, the probability that they all occur on edge e at time t is at most $\left(\frac{1}{C}\right)^{\tau}$, since packet delays are chosen independently;

otherwise the probability is 0. Thus, we have

$$\begin{aligned} p &\leq \binom{C}{\tau}\left(\frac{1}{C}\right)^{\tau} \\ &\leq \left(\frac{eC}{\tau}\right)^{\tau}\left(\frac{1}{C}\right)^{\tau} \\ &= \left(\frac{e}{\tau}\right)^{\tau}, \end{aligned}$$

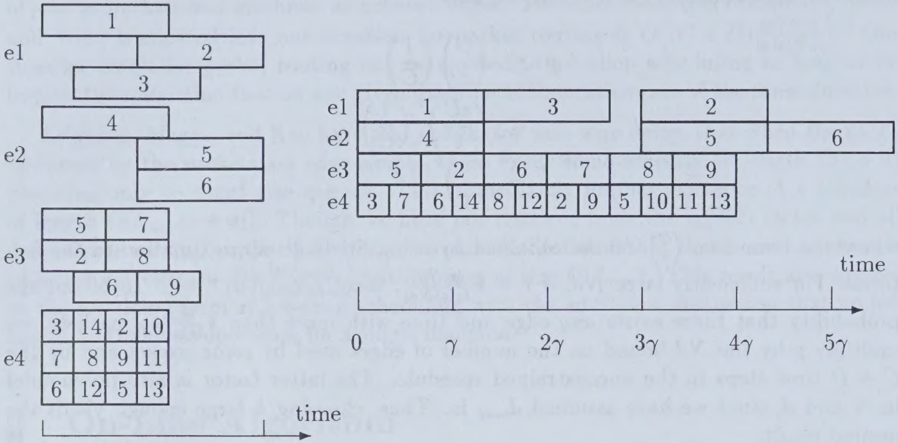
where the bound on $\binom{C}{\tau}$ can be obtained by using Stirling's approximation to the factorial. For sufficiently large Nd, if $\tau = k\frac{\lg(Nd)}{\lg\lg(Nd)}$, then, $p \leq (Nd)^{-(k-1)}$. To bound the probability that there exists *any* edge and time with more than $k\frac{\lg(Nd)}{\lg\lg(Nd)}$ packets, we multiply p by the Nd bound on the number of edges used by some packet and by the $C + D$ time steps in the unconstrained schedule. The latter factor is also polynomial in N and d, since we have assumed d_{max} is. Thus, choosing k large enough yields the desired result. ■

We must now explain how to flatten the unconstrained schedule into a legitimate schedule. The flattening procedure is trivial when each wire delay is just one unit of time. In that case, an unconstrained schedule S of length L with at most γ packets on an edge at one time can be flattened to a legitimate schedule of length γL by replacing each unit of S's time with γ units of time in which the packets on any given edge are routed in turn. In the more general context of nonunit wire delay, Shmoys, Stein, and Wein show how to flatten into a schedule of length $\gamma L \lg d_{max}$, but we show that a flattened schedule of length γL can be obtained by taking advantage of a distinction between packet routing and the general form of job-shop scheduling considered in [4]. In particular, any two packets that traverse the same edge spend the same amount of time in transit on that edge. We further show that we can flatten schedules produced as in Lemma 1 on-line, whereas Shmoys, Stein, and Wein consider only an off-line context.

Lemma 2 *Consider any unconstrained schedule of length L with at most γ packets on an edge during any time step. The unconstrained schedule can be simulated by γL steps of a legitimate schedule, and the simulation can be performed on-line if all the delays in the unconstrained schedule are in the initial queues.*

Proof. The flattening process involves routing the packets on each edge e in order of their start times for traversing e in the unconstrained schedule (with ties broken arbitrarily). Each packet is routed as soon as possible in the legitimate schedule (given the constraint of one packet per edge at any time), except that a packet that begins traversing edge e at time t in the unconstrained schedule does not do so before time γt in the legitimate schedule. (Figure 1 shows an example of an unconstrained schedule and its flattened version.)

The process just described can be accomplished on-line by having each packet carry a field that holds the time that the packet begins traversing the upcoming edge in the unconstrained schedule. Initially, each packet holds the delay assigned in its source processor. Each time a packet is dispatched on an edge, it adds in the delay of that edge. Thus, each network node can decide when to dispatch packets on its outgoing edges by inspecting the packets and associated information queued at its incoming edges.



(a) (b)

Figure 1: An example of the flattening process for four edges, e_1, e_2, e_3, and e_4. (a) The initial unconstrained schedule of $L = 7$ and $\gamma = 3$. (b) The flattened version of the schedule in (a).

It remains to be shown that all packets are routed by time γL. Let us refer to the routing of a particular packet on a particular edge as an *operation*. Also, let t^{ω}_{UB} and t^{ω}_{UE} represent the begin and end times for operation ω in the unconstrained schedule, and let t^{ω}_{LB} and t^{ω}_{LE} represent the times in the legitimate schedule. Our flattening process, clearly enforces $t^{\omega}_{LB} \geq \gamma t^{\omega}_{UB}$, and we now show that $t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$.

We proceed by induction on time. Under the assumption that $t^{\omega}_{LE} \leq \gamma t^{\omega}_{UE}$ whenever $t^{\omega}_{UE} \leq t$, we can show that the same is true whenever $t^{\omega}_{UE} \leq t+1$. (The base case with $t = 0$ is trivial.) Consider any operation ω with $t^{\omega}_{UE} = t+1$, and denote the packet and edge involved as m and e, respectively. (If there is no such operation, we are done.) Since $t^{\omega}_{UB} \leq t$, we know from the induction hypothesis that any operation ω' for a packet m' on edge e with $t^{\omega'}_{UB} \leq t^{\omega}_{UB}$ has the property that all other operations for m' on edges that precede e in the path of m' complete by time $\gamma t^{\omega'}_{UB}$ in the legitimate schedule. This implies that there exists such an operation ω' with $t^{\omega'}_{LB} = \gamma t^{\omega'}_{UB}$; we let ω^* be the one maximizing $t^{\omega^*}_{LB}$. By the definition of ω^*, the legitimate schedule places some packet on edge e at every time step from $t^{\omega^*}_{LB}$ to t^{ω}_{LE}. All of these packets begin at times in $[t^{\omega^*}_{UB}, t^{\omega}_{UB}]$ in the unconstrained schedule and therefore end by time t^{ω}_{UE} (since all traversals of edge e take time d_e). Thus the completion time of ω in the legitimate schedule is

$$\begin{aligned} t^{\omega}_{LE} &\leq t^{\omega^*}_{LB} + \gamma(t^{\omega}_{UE} - t^{\omega^*}_{UB}) \\ &= \gamma t^{\omega^*}_{UB} + \gamma(t^{\omega}_{UE} - t^{\omega^*}_{UB}) \\ &= \gamma t^{\omega}_{UE} . \end{aligned}$$

■

By putting together Lemmas 1 and 2, we obtain the following result:

Theorem 3 *When d_{max} is bounded above by a polynomial in N and d, any set of packets can be routed on-line in $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ steps using queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$, with high probability.* ■

We can remove the restriction on d_{max} by using a technique similar to [4], but again we must show how to perform the task on-line:

Theorem 4 *Any set of packets can be routed on-line in $O\Big((C+D)\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$ steps using queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$, with high probability.*

Proof. We can begin by thinking of each d_i as being rounded down to the nearest multiple of $\frac{d_{max}}{Nd}$, denoted d_i'. In the resultant network, $\mathcal{N}'$, edges have at most Nd distinct lengths which are multiples of $\frac{d_{max}}{Nd}$. By working with a routing clock period of $\frac{d_{max}}{Nd}$, we can use Lemma 1 to produce the unconstrained schedule we used above, since C and D are polynomial in N and d when expressed in units of $\frac{d_{max}}{Nd}$. The only problem is that in the real network $\mathcal{N}$, each d_i' must be adjusted upward to d_i. But each adjustment is at most $\frac{d_{max}}{Nd}$, which we can handle by simply doubling the clock period to $\frac{2d_{max}}{Nd}$, i.e., giving packets twice as much time at each step to travel or wait on the same edge as before. This adjustment does not change the number of packets using any edge during any time step, so we can proceed with the flattening process just as before. ■

We can also improve the schedule length by tightening the analysis in Lemma 1:

Theorem 5 *With high probability, on-line routing of any set of packets can be achieved in $O\Big(\frac{1}{\epsilon}(C\lg^{\epsilon}(Nd)+D\lg(Nd))/\lg\lg(Nd)\Big)$ steps using queues of size $O\Big(\frac{\lg(Nd)}{\lg\lg(Nd)}\Big)$.*

Proof. We modify Lemma 1 to produce an unconstrained schedule of length $D+\alpha C$ (for α to be determined) by choosing delays from $[1,\alpha C]$. Once τ is determined, the final flattened schedule will be of length $(D+\alpha C)\tau$. In Lemma 1, the upper bound on p becomes $\left(\frac{e}{\alpha\tau}\right)^{\tau}$. Then for $\alpha=(\lg(Nd))^{\epsilon-1}$ (with $\epsilon>0$) and $\tau=\frac{k}{\epsilon}\frac{\lg(Nd)}{\lg\lg(Nd)}$, we obtain $p\le(Nd)^{-(k-1)}$ for sufficiently large Nd. ■

It should be noted that the constant k in the proof of Theorem 5 is of modest size, so we certainly obtain a practical algorithm for $\epsilon=1$, the case that leads to Theorem 4. Even Theorem 4 specialized to unit edge delay improves upon the on-line result of Leighton, Maggs, and Rao except when c is somewhat larger than d. But we can also handle this case by obtaining an on-line algorithm with running time more closely parallel to that of Leighton, et. al. We could then interleave different routing algorithms to obtain an algorithm with running time on the order of the minimum of the running times of the individual algorithms.

Theorem 6 *Any set of packets can be routed on-line in $O(C+D\lg(Nd))$ steps using queues of size $O(\lg(Nd))$, with high probability.*

Proof. The proof is the same as for Theorem 5 except that we use $\alpha=1/\lg(Nd)$ and $\tau=\Omega(\lg(Nd))$. ■

3 Off-Line Schedule

In this section, we show that for any set of packets with edge-simple paths, there exists a schedule of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ using queues of size $O(d_{max})$. Our proof is nonconstructive, but the result may provide useful information. For a communication pattern that is to be used often enough, it may be worthwhile to spend substantial off-line computation time determining a good schedule.

In this section, we make heavy use of the Lovász Local Lemma [5]:

Lemma 7 (Lovász Local Lemma) *Let $A_1, \cdots, A_m$ be events each occurring with dependence at most b, i.e., every one of the events is mutually independent of at least $m - b$ other events. If $\forall i \Pr\{A_i\} \leq p$ and $4pb < 1$, then the probability that none of these events occurs is greater than zero.* ■

Our strategy to obtain the off-line schedule consists of three stages. In the first stage, we use an approach similar to Leighton, Maggs, and Rao [2, 3] of making a succession of refinements to the "greedy" schedule, in which packets never wait. In this succession of refinements, we bound the congestion in smaller and smaller intervals of time until the number of packets using an edge is at most $O(d_{max})$ during any set of $\Theta(d_{max}^2)$ consecutive time steps. In the second stage, we bound the number of packets using an edge during any unit time step to $O\Big(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$. In the final stage, we produce a legitimate schedule by applying the flattening process described in Section 2.

We only sketch the iterative refinement process of the first stage, since it closely parallels that of Leighton, et. al. First, we review a few definitions. A set of T consecutive time steps is referred to as a *T-frame* or a *frame of size T*. We also define the congestion in a frame to be the largest number of packets that traverse any edge during the frame. The *relative congestion* in a frame is the ratio of the congestion in the frame to the size of the frame. We begin with a special refinement that transforms the greedy schedule, S_0, into a schedule S_1 in which the relative congestion in each $(d_{max} \lg c)$-frame is $O\Big(\frac{1}{d_{max}}\Big)$. Each successive refinement transforms a schedule S_i with relative congestion at most r_i in any frame of size (at least) $d_{max} I_i$ (with $I_i = \Omega(d_{max})$) into a schedule S_{i+1} with relative congestion at most r_{i+1} in any frame of size (at least) $d_{max} I_{i+1}$, where $r_{i+1} \approx r_i$ and $I_{i+1} \ll I_i$.

For the initial refinement that produces S_1 from the greedy schedule of length $|S_0| = D$, we assign each packet an integral delay x chosen randomly and uniformly from the interval $[1, \alpha c d_{max}]$. The resultant schedule is of length $\alpha c d_{max} + D$. Without loss of generality, we assume that $cd_{max} = D$, so $|S_1| = (1 + \alpha) c d_{max}$. We can also assume that $c = \Omega(d_{max})$; otherwise we skip this stage and apply Lemma 9 directly to the greedy schedule.

We claim that there is some way of choosing x's so that for $I_1 = \lg c$, the relative congestion is $O\Big(\frac{1}{d_{max}}\Big)$ in any frame of size (at least) $I_1 d_{max}$ in S_1. We prove this claim by using the Lovász Local Lemma. For each edge e, we define a bad event as the event that more than $\lg c$ packets use e during any $(d_{max} \lg c)$-frame. For any packet, there are at most $d_e \leq d_{max}$ delays that cause it to be present on e at a particular time step. Thus, the probability that it appears on e during a particular $(d_{max} \lg c)$-frame is at most

$\frac{d_{max}\lg c + d_{max}}{\alpha c d_{max}}$. So, the probability that any bad event occurs is

$$p \leq (1+\alpha)cd_{max}\binom{c}{\lg c}\left(\frac{2d_{max}\lg c}{\alpha c d_{max}}\right)^{\lg c},$$

since the number of $(d_{max}\lg c)$-frames is certainly at most $(1+\alpha)cd_{max}$. Also, at most c packets pass through an edge, and each of these packets passes through at most $d \leq D$ other edges. Therefore, since we were able to assume $cd_{max} = D$ and $c = \Omega(d_{max})$, we have a dependence b of at most $cD = O(c^3)$. Thus, for a sufficiently large constant α, we have $4pb < 1$, from which the claim follows for frames of size *exactly* $I_1 d_{max}$. The proof is easily extended to apply to all frames of size *at least* $I_1 d_{max}$, and we henceforth do not make this distinction.

We now sketch the remainder of the refinement process. The first step in the ith refinement is to break S_i into blocks of $(2I_i^3 + 2I_i^2 - I_i)d_{max}$ consecutive time steps and reschedule each block independently.

Within each block, we assign each packet an integral delay x chosen randomly and uniformly from $[1, I_i d_{max}]$. A packet assigned delay x is actually delayed once every I_i steps in the first xI_i steps. In order to make the packets end up in the same positions at the end of the rescheduled block as in the block of S_i (so that the blocks remain independent), we also insert a delay every I_i steps in the last $(I_i d_{max} - x)I_i$ steps. Since we are allowing nonunit edge delays, some of the delays we have inserted may occur in the midst of an edge traversal rather than at a queue, but we will show how to move the delays to queues later. If $2I_i^3 + 2I_i^2 - I_i < I_{i-1}$, which holds as long as I_i is larger than some constant, a packet has been delayed at most once in the entire block before we insert new delays. If any new delay would be within I_i steps of the single old delay, it is not inserted. Omitting these delays has a negligible effect on the probability analysis, but it allows us to maintain the invariant that every packet waits at most once every I_i steps in S_{i+1}.

It can be shown that there is some way of choosing the delays so that in between the first and last $I_i^2 d_{max}$ steps of a block, we can decrease the frame size from $I_i d_{max}$ to $(\lg^2 I_i)d_{max}$ without increasing the relative congestion much. Furthermore, we can keep the relative congestion from increasing much in the first and last $I_i^2 d_{max}$ steps by increasing the frame size in these regions from $I_i d_{max}$ to $I_i^2 d_{max}$. Now, we move the block boundaries so that the regions with the larger frame size form a "fuzzy region" at the center of each new block. Finally, we decrease the frame size in the fuzzy regions with little increase in the frame size elsewhere through another round of assigning random delays to packets. This time packets choose integral delays randomly and uniformly from $[1, I_i^2 d_{max}]$. A packet with delay x waits once every $(I_i^3 d_{max}/x)$ steps in the first $I_i^3 d_{max}$ steps (i.e., before the fuzzy region) and once every $I_i^3 d_{max}/(I_i^2 d_{max} - x)$ steps in the last $I_i^3 d_{max}$ steps (after the fuzzy region).

Lemma 8 summarizes the result of this refinement procedure:

Lemma 8 *As long as $I_i = \Omega(d_{max})$, the ith refinement step described above decreases the frame size from $I_i d_{max}$ to $I_{i+1}d_{max} = (\lg^4 I_i)d_{max}$ for the entire schedule, while the relative congestion becomes $r_{i+1} = r_i(1 + O(1)/\sqrt{\lg I_i})$.*

Proof. This lemma can be proved in a similar fashion to [2, 3]. ■

We perform refinement steps until we obtain a schedule S_j with $I_j = O(d_{max})$. In S_j, the congestion in frames of size $O(I_j d_{max}) = O(d_{max}^2)$ is $O(I_j) = O(d_{max})$. At this point, we move delays that fall in the midst of an edge traversal to the edge queue for the corresponding edge, which has no effect on the congestion in any frame and does not require queues larger than $O(d_{max})$. Since every packet waits at most once every $I_{j-1} = \Omega(d_{max})$ steps in S_j, we end up with each packet waiting at most once in each edge queue. Thus, we have obtained a schedule S^* of length $O(cd_{max})$ and congestion $O(d_{max})$ in each $O(d_{max}^2)$-frame.

Now we proceed to the second stage of our off-line construction, which we apply to frames of size $\Theta(d_{max}^2)$ that have congestion $O(d_{max})$. (Recall that the refinement stage actually bounds the relative congestion in all frames of size at least T for some T that is $O(d_{max}^2)$.) The required result for the second stage is embodied in the following lemma:

Lemma 9 *Suppose we have a schedule S_U with congestion $k = O(d_{max})$ and $|S_U| = O(kd_{max})$. Then, there is a schedule, of length $O(kd_{max})$, in which at most $O\Big(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ packets use an edge during any unit time step.*

Proof. Each packet chooses a delay x randomly and uniformly from $[1, \alpha k d_{max}]$, where α is a constant to be determined. The resultant schedule is of length $\alpha k d_{max} + |S_U| = O(kd_{max})$.

We claim that for $\eta = \Omega(\frac{\lg(d_{max})}{\lg\lg(d_{max})})$, there is a way of choosing x's such that at most η packets use an edge during any unit time step. Again we use the Lovász Local Lemma. For each edge, the bad event of more than η packets using the edge at some time step occurs with probability p at most $O(kd_{max})\binom{k}{\eta}\left(\frac{d_{max}}{\alpha k d_{max}}\right)^{\eta}$. The dependence b is at most $k\,|S_U| \leq O(d_{max}^3)$. Thus, for a $\alpha \geq e$ and $\eta = \Omega(\frac{\lg(d_{max})}{\lg\lg(d_{max})})$, we have $4pb < 1$, from which the claim follows. ■

To complete the second stage we use Lemma 9 to independently reschedule frames of size $\Theta(d_{max}^2)$ with congestion $O(d_{max})$ in S^*. (We are able to make the frames independent by extending each one by d_{max} time steps so that each operation of routing a packet on an edge occurs entirely within one frame.) We end up with a schedule of length $O(cd_{max})$ with at most $O\Big(\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ packets on an edge at any time step and queues that still have size $O(d_{max})$.

Finally, for the third stage of the off-line construction, we simply apply Lemma 2, and we have proved the following theorem:

Theorem 10 *For any set of packets with edge-simple paths, there exists a legitimate schedule of length $O\Big((cd_{max} + D)\frac{\lg(d_{max})}{\lg\lg(d_{max})}\Big)$ using queues of size $O(d_{max})$.* ■

References

[1] F. T. Leighton, Bruce M. Maggs, Abhiram G. Ranade, and Satish B. Rao. Randomized routing and sorting on fixed-connection networks. Manuscript, 1991.

[2] F. T. Leighton, Bruce M. Maggs, and Satish B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. Manuscript, 1991.

[3] Tom Leighton, Bruce Maggs, and Satish Rao. Universal packet routing algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pages 256–269. IEEE Computer Society Press, 1988.

[4] David B. Shmoys, Clifford Stein, and Joel Wein. Improved approximation algorithms for shop scheduling problems. In *Proceedings of the 2nd Annual SIAM Symposium on Discrete Algorithms*, pages 148–157, 1991.

[5] Joel Spencer. *Ten Lectures on the Probabilistic Method.* SIAM, 1987.

Communications Networks as Stochastically Perturbed Nonlinear Systems: A Cautionary Note

Robert S. Maier*
Department of Mathematics,
University of Arizona,
Tucson, Arizona 85721, U.S.A.

Abstract

Lim, Meerkov *et al.* [4, 3, 9] have studied large homogeneous communications networks from a stochastic point of view. If a Markovian access control discipline is used, the evolution of an M-node network, in which each node buffers up to N packets, is in the $M \to \infty$ limit modelled as a random walk (with drift) in an N-dimensional polyhedron. This random walk may be viewed as a stochastically perturbed nonlinear system. If this system has more than a single point of stable equilibrium, transitions between them will occasionally occur. Lim and Meerkov have modelled this phenomenon as a one-dimensional first passage time problem. We show, by considering a related model, that one-dimensional approximations of this sort can result in arbitrarily large errors.

1 Introduction

It is often possible to model a communications network, or a single computer system, *stochastically*: as a finite-state Markov chain. However the number of states is typically very large. Fortunately many Markov chains that arise in system modelling are *lumpable*: the system state space can be replaced by a much smaller reduced state space, by merging states. In a series of papers, Lim, Meerkov and coworkers [4, 3, 9] have studied large homogeneous communications networks from this point of view. If time-slotted Markovian access control disciplines are used, a Markov chain model is appropriate, and the reduced state space Q is naturally viewed as a a set of integer-valued N-tuples, for some N. The identification is as follows. If there are M nodes on the network, and each node can exist in $N + 1$ internal states (the possible occupancies of a queue of maximum length N), then a reduced state would be of the form $\mathrm{h} = (h_1, \ldots, h_N)$, an N-vector. h_k is the number of nodes buffering at least k packets. Necessarily $M \geq h_1 \geq \cdots \geq h_N \geq 0$.

When taking the limit of a large network, it is useful to consider a *normalized* reduced state space $M^{-1}Q$, *i.e.*, $\{ M^{-1}\mathrm{h} \colon \mathrm{h} \in Q \}$, which is a discrete subset of the polyhedron

$$\mathcal{Q} \triangleq \{ \mathrm{y} \in \mathbb{R}^N : 1 \geq y_1 \geq \cdots \geq y_N \geq 0 \}. \tag{1}$$

---

*Supported in part by the National Science Foundation under grant NCR-9016211.

Lim, Meerkov *et al.* have observed that if the parameters of a network with a single collision channel (*i.e.*, packet arrival rates and transmission probabilities) are scaled appropriately as $M \to \infty$, the normalized chain $\mathbf{y}(1), \mathbf{y}(2), \ldots$, taking values in $\mathcal{Q}$ can be viewed as a *slow random walk*: to leading order, a deterministic mean drift (the "fluid approximation" to the chain), subject to small (as $M \to \infty$) stochastic perturbations which cause it, on a long time scale, occasionally to move against the flow. If the deterministic drift, viewed as a vector field on $\mathcal{Q}$, has stable equilibrium points (*i.e.*, the network as a whole has stable congestion points), the system's stationary density will be tightly concentrated around them. But such equilibrium points will in general only be *metastable*: on account of stochasticity, the system will occasionally move from one of them to another.

When $N = 1$ the reduced state space is one-dimensional, and it is well understood how to compute the large-M asymptotics of the transition rate between metastable states. A variety of rigorous and nonrigorous methods for dealing with one-dimensional first passage time problems are known: Wentzell-Freidlin theory [13], the WKB expansions of Matkowsky, Schuss, Knessl *et al.* [7, 11], and the approach of the present author [10], which draws extensively on the variational formulation of classical mechanics. Lim, Meerkov *et al.* have treated the $N = 1$ model in great detail [3, 9].

The $N > 1$ problem is much harder. Methods for computing the full (as opposed to leading) asymptotics of multidimensional jump process first passage times are not yet well developed. Lim, Meerkov *et al.* view the drift field on $\mathcal{Q}$ (when $N = 2$) as a *nonlinear dynamical system*, and have observed that in the large-M limit the system state, when jumping between congestion points, tends to pass through a hyperbolic equilibrium point, or *saddle*: a point where the vector field equals zero, and where the two eigenvalues of its linearization have opposite sign. In fact the normalized system state as a function of time, during the transition, tends to follow the unstable manifold of this hyperbolic point. This manifold (one-dimensional, of course) can be computed numerically. That being the case, they view the $N = 2$ Markov chain as *effectively one-dimensional*, and compute the large-M asymptotics of the transition rate between metastable states by the formulæ of Matkowsky, Schuss, Knessl *et al.*

In this note we argue that the one-dimensional approximation is unwise, since it is not possible to ignore the effects of transverse fluctuations: small motions of the system state away from the unstable manifold. These fluctuations do not "average out," at least not in any universal way. The behavior of the drift field in a neighborhood of the manifold is accordingly very important to the large-M asymptotics of the transition rate. Approximating the motion along the manifold by a one-dimensional motion may give rise to arbitrarily large errors in the computed mean first passage time.

We show this in the context of a much simplified model of a stochastically perturbed nonlinear system, which (since it uses continuous time rather than discrete, and has a continuous state space) can be viewed as a diffusion approximation. It is of course well known (see, *e.g.*, Kushner [8]) that the taking of a diffusion approximation may affect first passage time asymptotics. But there is no reason to believe that transverse fluctuations are any less important for discrete systems than for continuous ones. We analyse the idealized continuous-time model here because of its simplicity; a treatment of the full model of Lim, Meerkov *et al.* is under way.

In Section 2 we review the network model, and explain how it can be viewed as a stochastically perturbed dynamical system. In Section 3 we introduce our continuous-time model, and compute the asymptotics of the transition rate between its metastable states. For this we use a technique recently introduced by Naeh *et al.* [12]. Conclusions appear in Section 4.

2 The Network Model

The following description is only a brief sketch; see [4] for more. We assume that to every node on the network is attached an independent input process, generating a stream of packets. Each such process is modelled as a Bernoulli sequence with parameter p_0. So during each time slot, a packet arrives at each node with probability p_0. Since there is assumed to be only a single transmission channel, we take $Mp_0 \leq 1$ for stability.

Transmission of a packet proceeds as follows. In any time slot, there is some h_1-dependent probability $P_{\mathrm{tr}}(h_1)$ that exactly one of the h_1 blocked nodes will successfully transmit. $P_{\mathrm{tr}}(h_1)$ depends strongly on the channel access control discipline. If for example an ALOHA protocol is used, each of the h_1 blocked nodes will attempt, with some probability p_1, a transmission at the beginning of every time slot. T, the number of packets transmitted, will be binomial with parameters h_1 and p_1. So in the ALOHA case, $P_{\mathrm{tr}}(h_1) = \mathsf{P}\{T = 1\} = h_1 p_1 (1 - p_1)^{h_1 - 1}$.

By combining the changes in h due to arrivals and to successful transmissions, the transition probabilities of the chain $\mathrm{h}(0), \mathrm{h}(1), \ldots$ may be computed. We write $\mathrm{h}(n+1) = \mathrm{h}(n) + \xi_n$, where ξ_n, $n = 0, 1, \ldots$ is a sequence of independent $\mathbb{Z}^N$-valued r.v.s. The ξ_n are best thought of as independent instances of a single $\mathbb{Z}^N$-valued r.v., whose density is parametrized, in a way that can be worked out explicitly, by the current system state $\mathrm{h}(n)$. It is easily checked that if the model parameters p_0 and p_1 scale with M according to $p_i = q_i/M$, in the large-M limit the density of this r.v. ξ will depend only on the current *normalized* state $\mathrm{y}(n)$, *i.e.*, $\mathrm{h}(n)/M$. We accordingly write

$$\mathrm{y}(n+1) = \mathrm{y}(n) + M^{-1}\xi(\mathrm{y}(n)). \tag{2}$$

It follows from (2) that the Markov chain $\mathrm{y}(n)$ is a *slow random walk*: a nonlinear system subject to small perturbations. If $\mathrm{u}(\mathrm{x}) \triangleq \mathsf{E}\xi(\mathrm{x})$ is the mean drift vector field on $\mathcal{Q}$, and $t \triangleq n/M$ is *normalized time*, then to leading order the solution of (2) will tend to follow the deterministic trajectories which are solutions of $\dot{\mathrm{y}} = \mathrm{u}(\mathrm{y})$. (This is the usual fluid approximation.) A better approximation is provided by the coupled Itô stochastic differential equations

$$dy_i(t) = u_i(\mathrm{y}(t))\,dt + M^{-1/2} \sum_j \sigma_{ij}(\mathrm{y}(t))\,dw_j(t) \tag{3}$$

in which the tensor field $\sigma_{ij}(\mathrm{y})$ must be chosen consistent with the covariance matrix of the components of $\xi(\mathrm{y})$.

Lim, Meerkov *et al.* have noted that for certain values of q_0 and q_1 the vector field u will have more than one point of stable equilibrium in $\mathcal{Q}$. (See, in the $N = 2$ ALOHA case, Figure 6 of [9].) Typically it will have two: one low-congestion point, where most nodes are unblocked, and one high-congestion point. Their basins of attraction will partition $\mathcal{Q}$, and a separatrix, or dividing surface, will separate the two basins. There is normally a hyperbolic point on the separatrix, and the separatrix will be its stable manifold. Its unstable manifold will be one-dimensional, and will pass through the two stable points.

By introducing new curvilinear coordinates $(x_1, \ldots, x_N)$ on $\mathcal{Q}$, we can arrange matters so that the two unstable states lie on the x_1-axis, the hyperbolic point lies at the origin, and the separatrix is the $x_1 = 0$ hyperplane. In the next section we consider a model similar to this standardization.

3 A Related Continuous Model[1]

Consider a diffusion process $\mathrm{x}(t)$ in $\mathbb{R}^2$, defined by the coupled Itô stochastic differential equations

$$dx_i(t) = u_i(\mathrm{x}(t))\, dt + \epsilon^{1/2}\, dw_i(t), \qquad i = x, y. \tag{4}$$

We assume that the two components $u_x(x, y), u_y(x, y)$ of the vector field $\mathrm{u}(x, y)$ are sufficiently smooth, and can be expanded near the x-axis as follows:

$$u_x(x, y) = v_0(x) + v_2(x)y^2 + o(y^2) \tag{5}$$

$$u_y(x, y) = u_1(x)y + o(y). \tag{6}$$

We assume moreover that $u_1(x) < 0$ for all x, and that $v_0(x)$ has only three zeroes: $x_s > 0$, 0, and $x'_s < 0$. $v_0(x)$ is assumed to take signs $+, -, +, -$ on the four intervals $(-\infty, x'_s)$, $(x'_s, 0)$, $(0, x_s)$, $(x_s, +\infty)$. With these conventions $(x_s, 0)$ and $(x'_s, 0)$ will be stable equilibrium points of the vector field $\mathrm{u}(x, y)$, the point $(0, 0)$ will be a hyperbolic point, and its unstable manifold will lie along the x-axis. We assume that the vector field has no other equilibrium points, and that the y-axis is the separatrix between the two basins of attraction.

This model is suggested by the diffusion approximation (3), expressed in terms of the appropriate curvilinear coordinates. The only real change is that for simplicity we have replaced tensor diffusion by scalar diffusion. When the parameter ϵ is small, $\mathrm{x}(t)$ will spend most of its time in the vicinity of the two points of stable equilibrium. But it will occasionally transit between them. We impose the initial condition $\mathrm{x}(0) = (x_s, 0)$, and denote by t_{exit} the first time t at which the separatrix is reached.

To compute the $\epsilon \downarrow 0$ asymptotics of the expectation of t_{exit}, we apply formal asymptotic methods to the forward Kolmogorov equation. The forward Kolmogorov operator is

$$\mathcal{L}^* \triangleq \frac{\epsilon}{2}\Delta - u_x(x, y)\partial_x - u_y(x, y)\partial_y - (\partial_x u_x(x, y)) - (\partial_y u_y(x, y)). \tag{7}$$

If the diffusion process has a stationary density $\rho^{(0)}(x, y)$ (we assume that it does) it will satisfy $\mathcal{L}^* \rho^{(0)} = 0$. We are actually interested in a slightly different density function: the *quasi-stationary* density $\rho^{(1)}(x, y)$. This is the 'slowest decaying mode' if we impose Dirichlet (absorbing) boundary conditions on the separatrix. It is the eigenfunction of $\mathcal{L}^*$, equipped with these boundary conditions, whose eigenvalue λ_1 has the greatest real part. λ_1 will be real and negative; also, it will converge to zero exponentially fast as $\epsilon \downarrow 0$. In this limit $\mathsf{E}t_{\text{exit}}$ will be well approximated by $(-\lambda_1)^{-1}$, the timescale on which the mode decays. $-\lambda_1$ is the rate, in equilibrium, at which transitions between x_s and x_s' take place.

---

[1] This section is based on work done in collaboration with D. L. Stein of the University of Arizona.

The following technique for computing the $\epsilon \downarrow 0$ asymptotics of λ_1 is due to Naeh *et al.* [12]; it relies on the approximation of $\rho^{(1)}(x,y)$ in three different regions. First, note that it is easy to approximate $\rho^{(1)}(x,y)$ in a neighborhood of the stable point $(x_s, 0)$; a Gaussian approximation is appropriate. (In fact away from a 'boundary layer' of width $O(\epsilon^{1/2})$ near the separatrix, $\rho^{(1)}(x,y)$ will closely resemble $\rho^{(0)}(x,y)$.) A similar approximation may be constructed near the hyperbolic point $(0,0)$, by separation of variables. (Up to exponentially small errors, we may approximate $\rho^{(1)}(x,y)$ there by a solution of $\mathcal{L}^*\rho^{(1)} = 0$ which satisfies the Dirichlet boundary conditions.) And in the intermediate region, *i.e.*, in a neighborhood of the x-axis between $(0,0)$ and $(x_s,0)$, we may use a WKB approximation. By adjusting normalization coefficients, these three solutions may be made to match.

Once a mutually consistent set of approximations to $\rho^{(1)}(x,y)$ has been obtained, $-\lambda_1$ may be computed as the normalized flux of probability through the separatrix, *i.e.*,

$$-\lambda_1 = \frac{\int_{-\infty}^{\infty} (\epsilon/2)\partial_x \rho^{(1)}(0,y)\,dy}{\int_{x\geq 0} \rho^{(1)}(x,y)\,dx\,dy}. \tag{8}$$

The numerator may be computed from the approximation near $(0,0)$, and the denominator (which is a normalization factor) from the Gaussian approximation near $(x_s,0)$. Contributions to the denominator from the other two regions are negligible.

We now implement this program. Let us write $\lambda_x(S)$ and $\lambda_y(S)$ for $v_0'(x_s)$ and $u_1(x_s)$ respectively; these are the eigenvalues of the linearization of the vector field $\mathbf{u}(x,y)$ at $(x_s,0)$. Similarly, write $\lambda_x(H)$ and $\lambda_y(H)$ for $v_0'(0)$ and $u_1(0)$. These are all negative except for $\lambda_x(H)$, which is positive.

In this notation the Gaussian approximation to $\rho^{(1)}(x,y)$ near $(x_s,0)$ will be

$$\rho^{(1)}(x,y) \sim \exp -\left(|\lambda_x(S)|(x-x_s)^2 + |\lambda_y(S)|y^2\right)/\epsilon. \tag{9}$$

By separating variables in the approximate Kolmogorov equation $\mathcal{L}^*\rho^{(1)} = 0$, the approximation to $\rho^{(1)}(x,y)$ near $(0,0)$ may be shown to be

$$\rho^{(1)}(x,y) \sim C\left[e^{z^2/4} y_2(\tfrac{1}{2}, z)\right]\Big|_{z=x\sqrt{2\lambda_x(H)/\epsilon}} \exp -|\lambda_y(H)|y^2/\epsilon. \tag{10}$$

Here $y_2(\frac{1}{2}, z)$ is the odd parabolic cylinder function of index $\frac{1}{2}$ defined in Abramowitz and Stegun [1], and C is a constant which will be chosen to make this approximation consistent with (9). This approximation is valid when $x = O(\epsilon^{1/2})$; equivalently, when the normalized coordinate $z = O(1)$. Note that as $z \to +\infty$, *i.e.*, in the 'far field,' the quantity in brackets has asymptotics $\sqrt{\pi/2}\, e^{z^2/2}$. Note also [1] that $y_2'(\frac{1}{2}, 0) = 1$, by definition.

In the intermediate region we assume the existence of a WKB solution of the form

$$\rho^{(1)}(x,y) \sim K(x,y) \exp -W(x,y)/\epsilon \tag{11}$$

which we approximate near the x-axis by

$$\rho^{(1)}(x,y) \sim K(x) \exp -\left(f_0(x) + f_2(x)y^2\right)/\epsilon \tag{12}$$

for certain functions $K(x)$, $f_0(x)$, $f_2(x)$. Substituting (12) into the approximate Kolmogorov equation $\mathcal{L}^*\rho^{(1)} = 0$ and equating powers of ϵ yields the following equations for these functions:

$$f_0' = -2v_0 \tag{13}$$

$$v_0 f_2' = 2u_1 f_2 + 2{f_2}^2 - 2v_0 v_2 \tag{14}$$

$$v_0 K' = (u_1 + f_2)K. \tag{15}$$

By (13) we may take

$$f_0(x) = 2\int_x^{x_s} v_0(x)\,dx. \tag{16}$$

In solving (14) and (15), it is helpful to recoordinatize the portion of the x-axis between $x = 0$ and $x = x_s$. Define a new coordinate s by $ds = -dx/v_0(x)$; $s = -\infty$ corresponds to $x = x_s$, and $s = +\infty$ to $x = 0$. In terms of s we have

$$\dot{f}_2 = -2u_1 f_2 - 2{f_2}^2 + 2v_0 v_2 \tag{17}$$

$$\dot{K}/K = u_1 + f_2 \tag{18}$$

the dot now signifying differentiation with respect to s. In order to match up the WKB solution (12) with the approximations (9) and (10) near the endpoints $s = -\infty$ and $s = +\infty$ respectively, we must choose the solution of (17) to satisfy $f_2(s = -\infty) = |\lambda_y(S)|$ and $f_2(s = +\infty) = |\lambda_y(H)|$.

It is difficult to solve the equations (17) and (18) in closed form, so we shall consider several important special cases. The first case is when $v_2 \equiv 0$. In this case the random process $\mathrm{x}(t)$ is effectively one-dimensional: to $O(y^2)$, near the x-axis the stochastic differential equation for $x(t)$ decouples from the equation for $y(t)$. (See (4)). It is not difficult to check that the solutions of (17) and (18) will be related by $K \propto {f_2}^{-1/2}$. So $K(x_s)/K(0)$ will equal $|\lambda_y(S)/\lambda_y(H)|^{1/2}$, and in order that the WKB solution (12) match up with the solutions (9) and (10) near the two stable points, we must have

$$C = \sqrt{2/\pi}\,|\lambda_y(H)/\lambda_y(S)|^{1/2} \exp -\frac{2}{\epsilon}\int_0^{x_s} v_0(x)\,dx. \tag{19}$$

(Here the $\sqrt{2/\pi}$ factor arises from the far-field asymptotics of the parabolic cylinder function y_2.) Substitution of (9) and (10) into the denominator and numerator of (8) respectively now yields

$$-\lambda_1 \sim \frac{1}{\pi}\sqrt{|\lambda_x(S)|\lambda_x(H)} \exp -\frac{2}{\epsilon}\int_0^{x_s} v_0(x)\,dx. \tag{20}$$

(In computing the numerator we have used $y_2'(\frac{1}{2}, 0) = 1$.) So

$$\mathrm{E}\,t_{\mathrm{exit}} \sim \frac{\pi}{\sqrt{|\lambda_x(S)|\lambda_x(H)}} \exp \frac{2}{\epsilon}\int_0^{x_s} v_0(x)\,dx. \tag{21}$$

Eq. (21) is the standard formula, in the theory of one-dimensional diffusion processes, for the small-noise asymptotics of the first passage time from a metastable state to an absorbing, characteristic boundary [12]. That is not surprising since this $v_2 \equiv 0$ case is effectively one-dimensional.

Another exactly soluble case is when the drift field u(x) is a gradient field, *i.e.*, is derived from some 'potential function' $\phi(x, y)$ by the formula $u(x) = -\nabla\phi(x)$. In this case v_2 will not normally be identically zero; in fact necessarily $v_2 = u_1'/2$. It is easy to check that the solutions of (17) and (18) will be $f_2(x) = -u_1(x)$ and $K(x) \equiv K$, some constant. Formula (8) will yield

$$\mathsf{E}\, t_{\text{exit}} \sim \frac{\pi}{\sqrt{|\lambda_x(S)||\lambda_x(H)}} \sqrt{\frac{\lambda_y(H)}{\lambda_y(S)}} \exp \frac{2}{\epsilon} \int_0^{x_s} v_0(x)\, dx \tag{22}$$

which is similar to the one-dimensional formula (21) but has a different constant multiplying the exponential factor. Formula (22) is well known as the generalization of (21) to the case of a two-dimensional drift field arising from a potential. It was first derived by Eyring [6] in the 1930's; see also Caroli *et al.* [2]. For a modern derivation, see Naeh *et al.* [12].

Lastly we consider the case when v_2 is taken to be an *arbitrary* function of x, not necessarily related to $u_1(x)$. In this case the integration of equation (18) with respect to s may yield a factor $K(x_s)/K(0)$ which is arbitrarily large or small. The corresponding value for C, and the pre-exponential factor in the formula for $\mathsf{E}t_{\text{exit}}$, may be completely different from what is seen in the formulæ (21) and (22). Only if the function $v_2(x)$ is specified is this constant factor determined.

We note that if $v_2(x)$ is taken to be arbitrary, an extraordinary phenomenon can occur. The function $f_2(x)$, which governs the width of the tube-like WKB solution (12), may vanish at some point along the x-axis. When this occurs, it can be regarded as a *focusing singularity*; for a numerical example, see Figure 1 of Day [5]. On the far side of the singularity, the form (12) must be generalized. Naively, if f_2 goes negative the tube bifurcates. We do not consider this possibility further here.

4 Conclusions

In the previous section we have seen that in our simplified model, the expected transition time $\mathsf{E}t_{\text{exit}}$ between metastable states grows exponentially in $1/\epsilon$. This corresponds, in the network context, to a rapid growth in M as $M \to \infty$. This assumes that the network parameters scale with M as described in Section 2. The rate of exponential growth depends only on the mean drift velocity on the unstable manifold of the hyperbolic point, which in our model we denote $v_0(x)$. It is computed by the same formula that is employed in solving one-dimensional small-noise first passage time problems.

But the pre-exponential factor in $\mathsf{E}t_{\text{exit}}$ depends also on the behavior of the mean drift in a neighborhood of the manifold. The function $v_2(x)$ was used in the preceding section to quantify this. It describes a sort of *differential shearing* present in most nonlinear dynamical systems: the small (quadratic) changes in the mean drift along the manifold due to small transverse fluctuations. It cannot be neglected when computing the first passage time asymptotics, for if nonzero it may affect the pre-exponential factor to an arbitrarily large extent. This occurs despite the fact that as $\epsilon \downarrow 0$, the system state is increasingly concentrated around the manifold.

We believe that this subtle effect provides an explanation for the discrepancies between Lim and Meerkov's analysis of the $N = 2$ network model [3, 9] and the results of their simulations [9]. Their multidimensional network model in a sense becomes one-dimensional as $M \to \infty$, but the one-dimensionality is not so strong as to permit an easy computation of the transition rate between metastable states.

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* Dover, New York, 1965.

[2] B. Caroli, C. Caroli, B. Roulet, and J. F. Gouyet. A WKB Treatment of Diffusion in a Multidimensional Bistable Potential. *J. Statist. Phys.* **22** (1980), 515–536.

[3] J. Cicero, J.-T. Lim, S. M. Meerkov, and Z. Schuss. Large Homogeneous Communications Networks with Markovian Access Control, II. Global Behavior and Residence Times. *J. Math. Anal. Appl.* **105** (1985), 84–103.

[4] J. Cicero, S. M. Meerkov, and Z. Schuss. Large Homogeneous Communications Networks with Markovian Access Control, I. Equilibria and Local Stability. *J. Math. Anal. Appl.* **103** (1984), 481–496.

[5] M. V. Day. Recent Progress on the Small Parameter Exit Problem. *Stochastics* **20** (1987), 121–150.

[6] S. Glasstone, K. J. Laidler, and H. Eyring. *The Theory of Rate Processes.* McGraw-Hill, New York, 1966.

[7] C. Knessl, B. J. Matkowsky, Z. Schuss, and C. Tier. An Asymptotic Theory of Large Deviations for Markov Jump Processes. *SIAM J. Appl. Math.* **46** (1985), 1006–1028.

[8] H. J. Kushner. A Cautionary Note on the Use of Singular Perturbation Methods for 'Small Noise' Models. *Ann. Probab.* **2** (1982), 42–50.

[9] J.-T. Lim and S. M. Meerkov. Theory of Markovian Access to Collision Channels. *IEEE Trans. Comm.* **COM-35** (1987), 1278–1288.

[10] R. S. Maier. The Instability Times of Multiaccess Broadcast Channels. In *Proceedings of the 28th Allerton Conference on Communication, Control and Computing*, pp. 139–148, Monticello, Illinois, 1990.

[11] B. J. Matkowsky, Z. Schuss, C. Knessl, C. Tier, and M. Mangel. Asymptotic Solution of the Kramers-Moyal Equation and First-Passage Times for Markov Jump Processes. *Phys. Rev. A* **29** (1984), 3359–3369.

[12] T. Naeh, M. M. Kłosek, B. J. Matkowsky, and Z. Schuss. A Direct Approach to the Exit Problem. *SIAM J. Appl. Math.* **50** (1990), 595–627.

[13] A. D. Wentzell. Rough Limit Theorems for Large Deviations for Markov Processes. *Theory Probab. Appl.* **21** (1976), 227–242, 499–512.

Multicast in ATM Networks

Matthew Doar
jmsd@uk.ac.cam.cl

University of Cambridge
Computer Laboratory [1]

Nigel G. Bean
N.G.Bean@uk.ac.cam.statslab

University of Cambridge
Statistical Laboratory [2]

ABSTRACT

The global telecommunications industry and its associated networks are undergoing preparations for massive changes in the next decade to provide faster and more reliable services, especially for video and data traffic. These Integrated Services Digital Networks (ISDN) will also depend upon the provision of many different forms of service, including the ability to have more than one destination in a connection *(multicast)*. This paper presents the design and performance studies of a small-scale communications switch which can provide such a multicast service. It has been found to be simple to implement and offers a performance comparable with more complex designs.

1 Introduction

Asynchronous Transfer Mode (ATM) networks have been proposed by the CCITT as the underlying basis for the future Broadband Integrated Services Digital Networks (B-ISDN) [1]. These networks are being designed to transport multimedia traffic and to be flexible enough to carry all future forms of traffic without massive hardware changes. Typical traffic will include data such as computer files, by its nature highly sensitive to errors in its transmission but relatively insensitive to delays, and real-time traffic such as voice and video, which can often tolerate higher error-rates but whose effectiveness may depend critically upon the transmission delay that it encounters. The scale of ATM networks can range from global to local, promising easier interoperability between private and public networks [2].

One of the available modes of communication in an ATM network will be multicast, specifically defined here as when the information from one source is sent to many destinations. Likely uses for multicast include any form of conferencing, entertainment video distribution or the interconnection of local area networks (LANs) using wider area ATM networks. Other uses of multicast which have been suggested are the interconnection of processors for distributed computing, which could be extended into the idea of building a workstation around an ATM switch [3].

Existing networks tend to either not provide multicast, or to provide it by restricting the inherent broadcast nature of their communication medium, so that destinations which are not supposed to receive the information filter it out. This wastes bandwidth compared to that necessary for the same effect in an inherently point-to-point ('unicast') network, though it does have the advantage of simplicity in many aspects. It also fails to scale well with the size of the network.

---

[1] University of Cambridge Computer Laboratory, New Museums Site, Pembroke Street, Cambridge CB2 3QG. U.K. Tel: +44 223 334618, Fax: +44 223 334679.

[2] University of Cambridge Statistical Laboratory, 16 Mill Lane, Cambridge CB2 1SB. U.K. Tel: +44 223 337958, Fax: +44 223 337956. Supported by The George Murray Scholarship and the Cambridge Commonwealth Trust.

To multicast efficiently, the replication of the information must take place at the lowest level of communication and as late as possible in the paths to the destinations. This paper describes one way of providing a small-scale low-level multicast service in an ATM network.

2 ATM Switches

B-ISDN ATM networks are based upon the idea of sending information in small fixed length blocks of data called 'cells', with 48 octets of data per cell, through interconnection points called switches. Each cell carries with it a two octet Virtual Circuit Identifier (VCI) to indicate which notional circuit the cell belongs to and hence to which destination a switch should route the cell. One of the attractions of ATM is that VCIs are assigned on a per switch basis, so that whilst their size remains constant, the use to which the VCI fields are put can differ from the unicast CCITT version. The multicast VCI for use inside the multicast ATM switch proposed in Section 4 comprises two octets. One octet is used as a multicast VCI, constant for each circuit. The other octet is divided into three fields; **Mcast** is a three bit pattern to distinguish multicast VCIs from unicast ones at the multicast fabric if no multicast VCI remapping occurs, **P** is a priority bit, indicating the priority of the cell and the four bit field **Fanout** is the required number of copies of this cell. Each copy is uniquely identified by labelling in the fanout field.

In an ATM switch with cascaded multicast capability (defined in Section 3), such as the one shown in Figure 1, the incoming VCI is mapped onto a multicast VCI at the multicast controllers and is then replicated within the multicast fabric. The multicast VCIs are then mapped onto their respective outgoing VCIs at the input port controllers and the copies of the cell are routed across the switch fabric to their respective output port controllers.

The design and construction of switch fabrics is now a relatively well understood problem, summarised for instance in Ahmadi and Denzel [4] and Tobagi [5]. The general goals are to minimise factors such as connectional complexity, the number of smaller switch elements which make up the switch fabric and the contention which can occur with some designs if several input ports wish to send cells to the same output port.

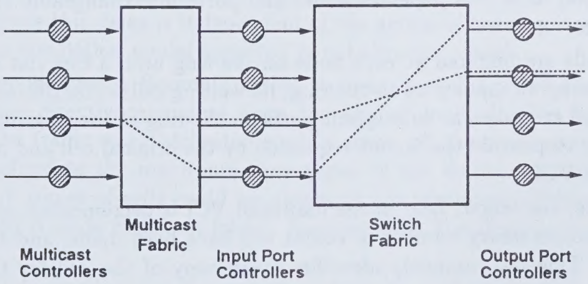


Figure 1: A 4×4 Multicast Switch

3 Multicast Fabric Designs

Multicast ATM switches proposed in the literature can be divided into two categories. In the first category there are multicast switch fabrics which have the ability to both replicate and route cells. In the second there are switches which have a multicast fabric prefixed (or *cascaded* [6]) to them which replicates the cells, but which leaves the routing to the switch fabric, such as the switch shown in Figure 1.

The first category of multicast switches can be constructed using an interconnection network where each switching element, often 2×2 input and outputs, can route an incoming cell to either one of the output ports or to both (Richards and Hwang [7]). Alternatively, they may use a ring both to replicate and route the cells (Rahnema [8], Forcina *et al* [9]). Lastly, a switch may be based upon shared memory, such as the Prelude [10] and Hitachi switches [11], where as many copies of the cell as required can be obtained by re-reading the contents of the cell buffer.

The multicast fabrics of the second category use the same three principles. Designs which do so are Bubenik and Turner [12], Lee [13] and Tarng and Meditch [14] (interconnection network), Newman and Doar [15] (ring), which is the design further expanded in Section 4, and Bianchini and Kim [16] (shared memory). The complexity of the first category, combining both the replication and routing functions in the multicast switch fabric has lead to further investigation of cascadable multicast fabrics due to their reduced overall complexity and increased modularity.

Of the many designs of ATM switches with multicast capability which have been proposed in the literature, only a few have actually been implemented. Some of these are the designs described in references [15, 16, 17, 18, 19, 20]. The next section gives further details of one of these [15].

4 A Slotted Ring Multicast Fabric

The multicast fabric design presented here is shown in Figure 2. It uses a uni-directional destination delete slotted ring to replicate the cells. Since the original proposal of the design [15], it has been modified as described in Section 6. Each slot on the ring is large enough to hold a complete cell (53 octets in ATM networks). The storage for the ring is provided within the ring nodes themselves (five octets) and in a FIFO buffer (not shown) which closes the ring and ensures that it contains an integer number of slots. The ring nodes are referred to in this paper as nodes and ports interchangeably and have input and output sides.

Incoming cells are buffered at each node on the ring until a free slot arrives. Upon successful access by an input port to the ring, its waiting cell is transferred into the slot. Copies of the cell are taken at subsequent ports on the ring as the ring rotates, as many as are necessary to provide the fanout requested by the original cell and now embedded in the multicast VCI.

At each node, the fanout field in the multicast VCI is decremented, so that when it reaches zero, the necessary number of copies will have been made, and the slot is free for use again. This also uniquely identifies every copy of the cell for the input port controllers, easing the problems of modifying multicast connections.

Since the same medium is used to transport and replicate cells, a node cannot be transmitting a cell and receiving a copy at the same time. This means that *fanout-1* nodes downstream of a transmitting node will be blocked from transmitting their waiting cells by the node already transmitting.

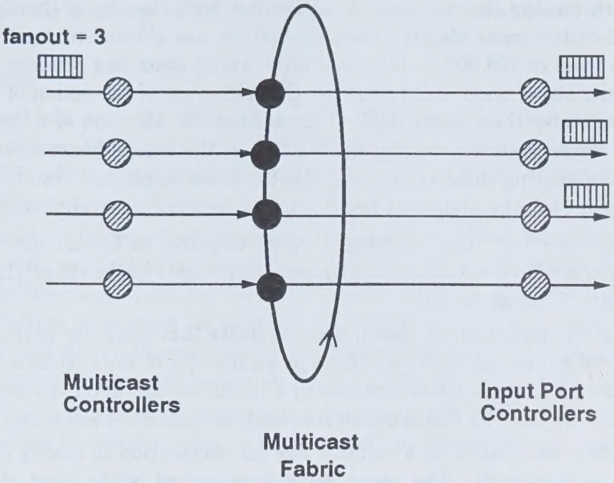


Figure 2: A 4×4 Multicast Fabric

The delay between individual copies of a cell arriving at the switch fabric for routing to their respective destinations is the rotation delay between nodes on the ring, which at a transmission rate of 155.52Mb/s and an 8 bit wide ring is less than $1\mu s$. It should also be noted that copies of a cell leave the multicast fabric on contiguous ports, which makes updating the VCI remapping tables in the input port controllers easier than designs such as Lee [13], where the exit ports depend upon the other traffic present in the multicast fabric at the time of replication.

5 Simulations

The performance of the ring based multicast fabric was investigated using a simulation model. All delay results are normalised to the time taken for a complete slot to pass one node. In reality there is a delay between the arrival of the head of a cell and the time the cell can be replicated, since a cell is completely buffered before it is available for transmission. Since this delay is independent of the action of the ring it was assumed to be fixed, i.e. the simulation model operated in cut-through mode[1].

The traffic model was a Bernoulli arrival process, modelling slotted traffic arriving with probabilities directly dependent upon the applied load, L. The first simulations assumed that the traffic was distributed equally across all the input ports. The applied load was normalised to the maximum throughput of the design, so that a load of 1.0 indicates that N copies of cells could be produced per revolution of the ring in a symmetrically loaded N node multicast fabric. Asymmetrical distributions are considered in Section 6.1.

In the initial simulations the phase relationship between the arrival slots and the head of the slot was constant and zero. A random phase relationship between the arrival slots and the head of slot means that an extra delay, with a mean of half a slot length, is added due to waiting for the next slot to arrive at a node. This delay was eliminated from the results by synchronising the slot arrivals to the ring, with the aim of showing the performance characteristics more clearly. Each simulation was allowed to reach stability and then run for a total of 100,000 transmitted cells, whilst observing Student t-distribution confidence limits based upon multiple runs. Consideration of the action of the simulator at high loads (greater than about 0.99) suggests that the duration of a 'busy cycle', the time between instances when no cells are waiting in the fabric, becomes excessive. The difficulties of completing sufficient busy cycles to obtain significant results in reasonable CPU time meant that the high load results should be considered approximate.

---

[1] Cut-through switches begin transmission of a message before the message has been fully received.

The fanout was either fixed at integer values co-prime to the number of nodes (see Section 6) or chosen from a uniform distribution of integers in the range $(1, N-1)$, with the mean fanout, $\overline{f}$, equal to $N/2$.

Some initial simulations (not shown here) indicate that the delay performance of the fabric is independent of the fabric size for a given number of slots (sufficient to reach all the nodes). The performance also depends in a linear fashion upon the number of slots. This was confirmed later by the approximate analysis presented elsewhere [21].

A 16×16 fabric was chosen as a suitable size for comparison to reality (see comments on scalability in Section 8). The mean delay from arrival of the head of a cell at the node until gaining access to the ring for transmission is shown by the graph on the left in Figure 3, for a fixed fanout of 15. The 99th percentile curve of the same delay and fanout are also shown in Figure 3, together with the results of the approximate analysis [21]. For lower values of fanout, the delays are smaller. Results for a more realistic uniformly distributed fanout, with mean fanout 8, are shown in the graph on the right of Figure 3 and are approximately comparable in value to those of a fixed fanout of 5.

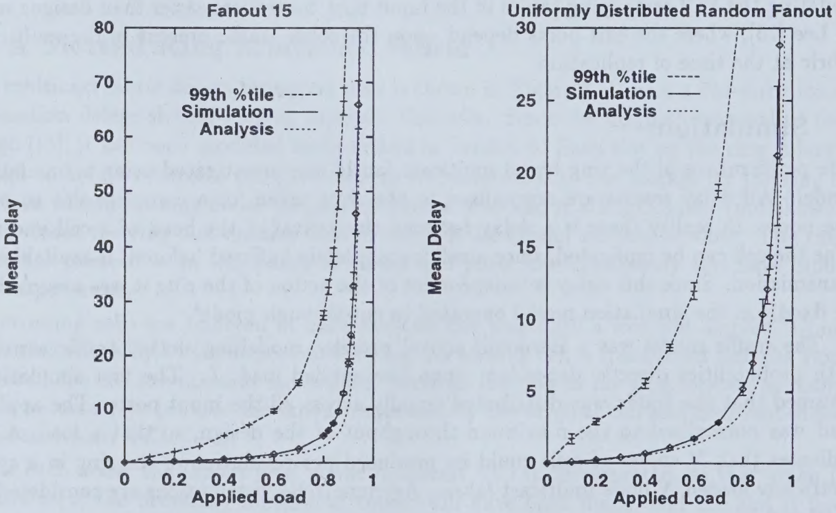


Figure 3: Mean Delay for Fixed and Uniformly Distributed Fanout

It may be seen from the graph on the right of Figure 3 that for an applied load of 0.7 the mean delay is 2.9 cell lengths and the 99th percentile delay is 18.8 cell lengths, corresponding to 7.9μs and 51.3μs at a line speed of 155.52Mbit/s. This is comparable to the delay across a unicast switch fabric operating at a similar load [12].

6 Fairness

The major problem with the design presented in Section 4 is that downstream input ports can be denied their fair share of the ring's bandwidth by a single highly loaded input port. This effect of asymmetrical loading is significant when either the applied load is high or the mean fanout is large.

The definition of fairness used here is simple. Each node should suffer approximately equal delays when sending cells through the multicast fabric, independent of the relative position of the incoming traffic on the fabric. Other parameters which might be considered in a definition of fairness include the input buffer queue size (and hence the Cell Loss Ratio at each node), the variance of the delay (jitter) and the 99th percentile of the delay as an approximate upper bound on the performance of the fabric.

Fairness can be improved by altering the distribution of traffic at the multicast fabric or denial of access to certain nodes. The next sections investigate the latter option.

6.1 Simulations

The performance of the multicast fabric without any denial of access was investigated as a basis for comparing different control algorithms. The total incoming traffic was assigned to each input port according to the bias parameter α, $(0 \leq \alpha \leq 1)$. The lightly laden ports all had their incoming traffic scaled by a factor $(1-\alpha)$ and the heavily loaded port's arrival rate was scaled by a factor $(1+\alpha(N-1))$, where N is the number of input ports. This distribution is referred to as the *point source distribution*. When $\alpha = 0$, the traffic was uniformly distributed across the multicast fabric and when $\alpha = 1.0$ all the traffic entered the fabric through a single port. It was assumed that the fanout distribution was the same for all input ports.

Figure 4 represents a 16 node fabric with the uniformly distributed random fanout (mean 8) mentioned in Section 5, at an applied load of 0.6. It can be seen that the effect of a single heavily loaded port is to increase the delay at certain downstream ports. As the number of cells emerging from a node decreases, so does the delay at that node. The crossover point comes after the ports are further away from the heavily loaded port than the mean fanout.

The overall effect of varying α and the applied load can be seen in the graph on the left in Figure 5. As α increases, the maximum throughput reaches a plateau at decreasing values until at $\alpha = 1$ the maximum throughput is equal to $\overline{f}/N$. Analytical results elsewhere [21] confirm the results shown here and show that for $L < 1/(1+\alpha)$ the throughput is equal to the applied load. This is a consequence of a necessary condition for the stability of the fabric.

The graph on the right in Figure 5 shows the difference between the mean delay (over ten runs) experienced by cells entering the multicast fabric at the most delayed port and the least delayed port, for example ports 1 and 15 in Figure 4. This value is a useful comparative measure of fairness. Whilst α is low, especially at low loads, the difference in the delays increases slowly. When $\alpha = 0$, the increase is due solely to fluctuations in the simulation results consequent upon finite simulation time. For lightly loaded switches ($\leq$ 0.4), the problems of unfairness, at least for the point source asymmetrical traffic model, do not appear to be excessive.

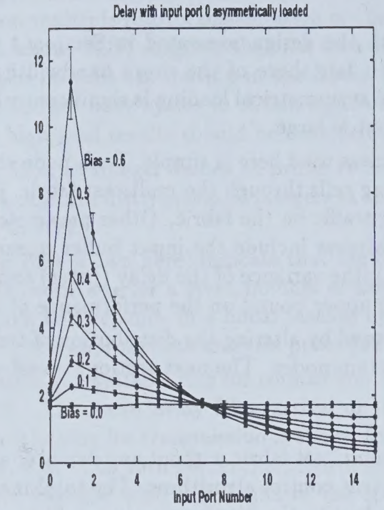


Figure 4: Mean Delay at Different Input Ports

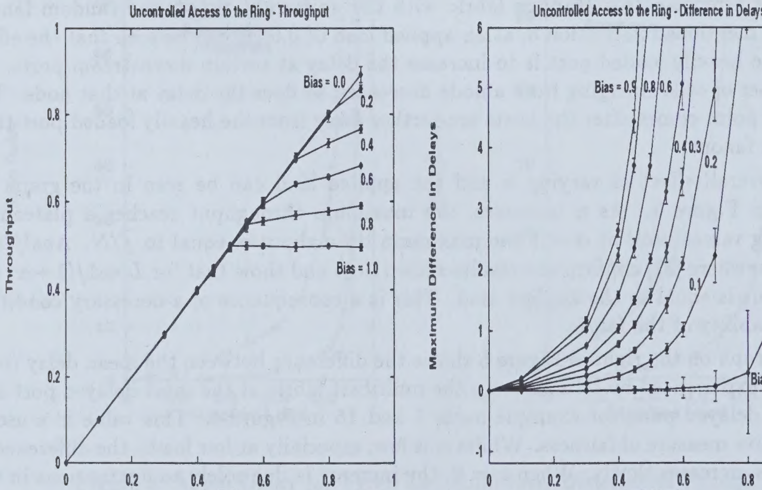


Figure 5: Unfairness in the Multicast Fabric

6.2 Simple Threshold Control Algorithm

The effect of one control access algorithm upon the throughput and fairness, measured as the percentage reduction in unfairness relative to the uncontrolled fabric, can be seen in Figure 6. The graphs in Figure 6 show the effects of the following algorithm:

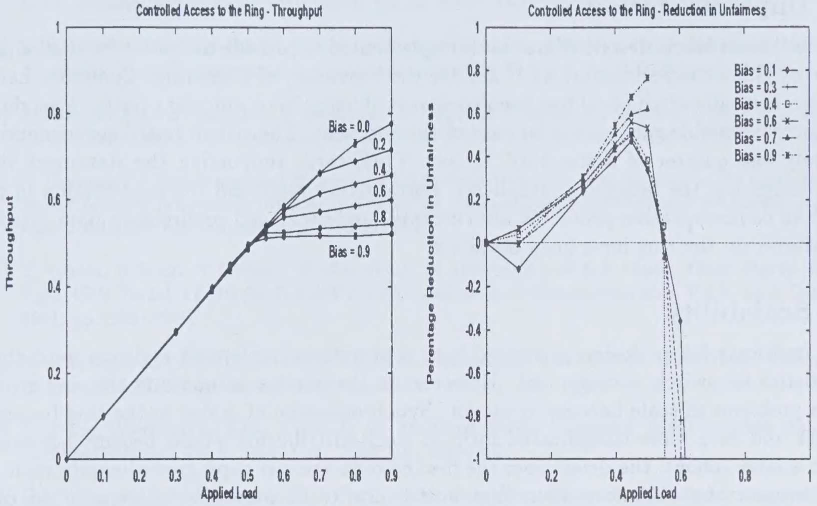


Figure 6: The Effect of the Simple Threshold Control Algorithm upon Unfairness

Each time a node, A, wishes to access the ring, if there is a cell at another node, B, which has been waiting longer than the threshold, R, slots *and* longer than the one at A *and* would be blocked by A's transmission, then node A is denied access.

This procedure is termed the Simple Threshold Control Algorithm (STCA). High values of R mean that the threshold is rarely reached and throughput and fairness results tend towards the uncontrolled results shown in Figure 5. Lower values of R reduce the maximum throughput to the same limit but for lower values of α, since the threshold is more often reached at lower loads and more of the ring's bandwidth is wasted. Since the multicast fabric might be expected to operate at a load of less than 0.6, a threshold was chosen to yield maximum performance below this load. This threshold was found to be $R = 2$ and the results are shown below.

The graph on the left in Figure 6 shows the maximum throughput. For $\alpha = 0.6$, for example, the maximum throughput has been slightly reduced from approximately 0.7 to approximately 0.6. The graph on the right in Figure 6 shows the percentage reduction in the fairness measure with and without the STCA. Confidence intervals have been omitted for the purposes of clarity and points where the fabric became unstable have also been omitted. It can be seen that the STCA improves the fairness up to a load where the maximum throughput is reached. The reduction in unfairness is especially marked for high values of α. For example, for $\alpha = 0.9$ and a load of 0.54, the maximum difference in delays is decreased from 4.42 to 1.43, a reduction of 68%.

One conclusion of these simulations is that the STCA does reduce unfairness in the multicast fabric, provided that the throughput has not reached its maximum value, which is slightly decreased by the presence of any control algorithm based upon denial of service. Another threshold control algorithm using independent thresholds for each node based upon the total output at the node was also investigated, but showed only a small improvement over the STCA and only for large α.

7 Implementation

The multicast fabric described has been implemented to provide multicast for the Fairisle network [22], an experimental ATM LAN in the University of Cambridge Computer Laboratory. A small switch (4×4) has been constructed using byte wide data paths throughout and programmable gate arrays for ease of development. The circuit board covers approximately one quarter of a standard 11 inch VME card, supporting the statement that this design has the benefit of simplicity. Performance tests and the modification of the existing communication protocols are currently underway and preliminary data rates of 100Mbit/s on the ring have been obtained.

8 Scalability

The multicast fabric design presented here is simple to implement and has good characteristics for switch management. However, as the number of nodes in the ring grows, some problems of scale become apparent. Synchronisation of nodes to the ring becomes harder and so a more complicated form of clock distribution would become necessary. With a large fanout, the delay from the first copy to the last copy grows linearly with N. The amount of state information that would have to be passed to implement the control algorithm simulated in Section 6 increases as N^2. Also, the design is most efficient when the number of bytes stored in the ring is just less than or equal to a multiple of 53. These issues, the convenient size of the fanout field in the multicast VCI and other economies of scale would suggest that a 32x32 ring-based multicast fabric would be the largest appropriate size using current technology.

9 Conclusions

A simple design for a small scale multicast fabric has been presented. Simulation and approximate analytic results suggest that its characteristics are comparable to other elements in ATM switches. A simple control mechanism has been simulated and found to reduce the unfairness in the multicast fabric, at the cost of reducing the maximum throughput. Other factors which affect the design of a multicast fabric have been noted and briefly examined.

References

[1] CCITT, Draft Recommendation I.150, *B-ISDN ATM Functional Characteristics*, Geneva, May 1990.

[2] D J Greaves and D McAuley, *Private ATM Networks*, Proceedings of 3rd International IFIP WG6.1/6.4 Workshop on Protocols for High-Speed Networks, Stockholm, May 1992.

[3] M D Hayter and D McAuley, *The Desk Area Network*, University of Cambridge Computer Laboratory, Technical Report 228, May 1991.

[4] H Ahmadi and W E Denzel, *A Survey of Modern High-Performance Switching Techniques*, IEEE Journal on Selected Areas in Communications, Vol.7, No.7, September 1989, pp.1091–1103.

[5] F A Tobagi, *Fast Packet Switch Architectures for Broadband Integrated Data Service Networks*, Proc. IEEE, Vol.78, No.1, January 1990, pp.113–167.

[6] C T Lea, *A New Broadcast Switching Network*, IEEE Transactions on Communication, Vol.36, No.10, October, 1988, pp.1128–1137.

[7] G W Richards and F K Hwang, *A Two-stage Rearrangeable Broadcast Packet Switching Network*, IEEE Transactions on Communications, Vol.33, No.10, October, 1985, pp.1025–1035

[8] M Rahnema, *The Fast Packet Switch: A High-Performance Efficient Architecture with Multicast Capability*, IEEE Transactions on Communications, Vol.38, No.4, April, 1990, pp.539–545

[9] A Forcina, T Di Stefano and E Taormina, *A Multicast Broadband Switching Module in a Hybrid ATM Environment*, Proceedings of International Conference Communications, 1989, (ICC '89)

[10] A Thomas, J P Coudreuse and M Servel, *Asynchronous time-division techniques: An experimental packet network integrating video communications*, Proceedings of International Switching Symposium 1984 (ISS '84), Florence, Italy, May 1984.

[11] T Kozaki, N Endo, Y Sakurai, O Matsubara, M Mizukami and K I Asano, *32x32 Shared Buffer Type ATM Switch VLSI's for B-ISDN's*, J. Selected Areas in Communications, Vol.9, No.8, October 1991, pp.1239–1247.

[12] R G Bubenik and J S Turner, *Performance of a Broadcast Packet Switch*, IEEE Trans. Communications., Vol.37, No.1, January, 1989, pp.60–69.

[13] T T Lee, *Nonblocking Copy Networks for Multicast Packet Switching*, IEEE J. Select Areas in Communications., Vol.6, No.9, Dec., 1988, pp.1455–1467.

[14] C L Tarng and J S Meditch, *A High Performance Copy Network for B-ISDN*, Proceedings of IEEE Infocom 1991, Miami, pp.171–177.

[15] P Newman and M Doar, *A Slotted Ring Copy Fabric for a Multicast Fast Packet Switch*, Proceedings of International Switching Symposium 1990 (ISS '90), Stockholm, Sweden, May, 1990.

[16] R P Bianchini, Jr. and H S Kim, *Design of a nonblocking Shared-Memory Copy Network for ATM*, Proceedings of IEEE Infocom 1992, Florence, pp.876–885.

[17] J R Cox and J S Turner, *Project Zeus Design and Application of Fast Campus Networks*, Technical Report 91-45, Department of Computer Science, Washington University, July 1991.

[18] ATMX, An Asynchronous Transfer Mode Switch for LAN Environments, Adaptive Corporation, announced July 1992.

[19] R J F deVries, *Switch Architectures for the Asynchronous Transfer Mode*, PhD Thesis, University of Twente, Netherlands, 1992.

[20] H S Kim and A Leon Garcia, *A Self-routing Multistage Switching Network for Broadband ISDN*, IEEE J. Selected Areas in Communications, Vol.8, No.3, April 1990, pp.459-466.

[21] N G Bean and M Doar, *Analysis of a Cyclic Server with Irregular Server Movements: A Model of an ATM Multicast Fabric*, in preparation.

[22] I M Leslie and D McAuley, *Fairisle: An ATM Network for the Local Area*, Proceedings of SIGCOMM '91, Zurich, September 1991.

Bilateral Control of Teleoperators with Time Delay Through a Digital Communication Channel

Gary M. H. Leung *   Bruce A. Francis*

Abstract

The stability of a teleoperator is considered when the communication channel between master and slave is implemented digitally and a communication time delay is incurred. A new control law for teleoperators has been introduced recently ([1]) which overcomes instability caused by the time delay in the communication channel. This method requires the assumption that both the master robot and slave robot are passive systems.

Since passive continuous-time systems, when implemented digitally via ideal sampler and zero-order hold are, in general, not passive, modifications are made to the control law introduced in [1] These modifications involve the introduction of three stable, linear, time-invariant, strictly causal filters. A procedure is given to determine a suitable sampling period h which ensures stability of the sampled-data master/slave system for any time delay in the communication channel.

---

*Systems Control Group, Dept. of Electrical Engineering, University of Toronto, Toronto, Canada. M5S 1A4

1 Introduction

We consider the standard teleoperator system depicted in Figure 1. The human operator commands a hand controller (the master system) by exerting a force f_h. The master system moves with velocity v_m which is transmitted via the communication channel to the slave. Likewise, the slave manipulator tracks the reference signal, v_{sd}, and the force sensed as a result of contact with the environment, f_e, is transmitted back to the master. The six signals shown in Figure 1 may be vector-valued.

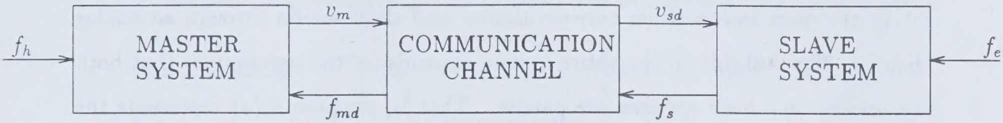


Figure 1: Standard Teleoperator System

The notation used in this paper will be as follows: in the time-domain, let P, T, ... denote operators and u, v, ... denote signals. In the frequency domain, let $\hat{E}$, $\hat{L}$, ... denote operators and $\hat{f}$, $\hat{g}$, ... denote complex-valued functions. Let $\varphi : u \mapsto \hat{u}$ be the Fourier transform. Corresponding to the time-domain operator P is the transfer function $\hat{p}$.

In 1966, Ferrell [2] first showed the instability due to time delay of a kinesthetically coupled teleoperator. Vertut et al. [3] showed that stability could be achieved in the presence of time delay only when the bandwidth of the system was severely reduced. However, Brooks [4] suggested that the bandwidth of a teleoperator should be approximately between 4 Hz to 10 Hz for normal operation. Hence, this problem had remained largely unsolved until recently.

In 1988, Anderson and Spong [1] introduced a new control law for teleoperators which overcomes instability caused by the time delay in the communication channel. It was shown that with a simple control strategy, the telerobotic system performs reasonably well for an ideal communication channel (no time delay). However, when a transmission delay exists, the system becomes unstable for certain inputs since the transmission delay leads to a non-passive communication channel. In order to remedy this problem, the equations of the communication channel were modelled to mimic a lossless transmission line. This has the advantage that the lossless transmission line equations model a passive two-port network regardless of the time delay. Subsequently in [5], asymptotic stability of master and slave velocities was proven.

2 Passivity and Communication Time Delay

In [1], the data transmission between master and slave occurs through an analog channel. The stability of the entire system depends on the assumption that both the master and slave systems are passive. That is, suppose $\hat{g}_s(s)$ represents the transfer matrix of the slave (or similarly the master) robot and

$$\hat{k}_s(s) = \begin{bmatrix} \hat{k}_{s11}(s) & \hat{k}_{s12}(s) \\ \hat{k}_{s21}(s) & \hat{k}_{s22}(s) \end{bmatrix}$$

represents the transfer matrix of the robot controller. The system can be represented as in Figure 2 where

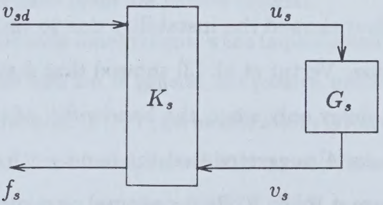


Figure 2: Slave System

$$\begin{bmatrix} f_s \\ u_s \end{bmatrix} = \begin{bmatrix} K_{s11} & K_{s12} \\ K_{s21} & K_{s22} \end{bmatrix} \begin{bmatrix} v_{sd} \\ v_s \end{bmatrix}$$

Passivity of this system means that for all inputs $v_{sd}(t)$ and corresponding outputs $f_s(t)$,

$$\int_0^\infty f_s'(t) v_{sd}(t) dt \geq 0$$

At this point, it is useful to state the following result (from e.g., [6]).

Lemma 1 *A stable system with transfer matrix $\hat{g}(s)$ is passive if and only if*

$$Re[\hat{g}(j\omega)] \geq 0 \quad \forall\omega$$

For single-input, single-output systems, the above lemma states that the Nyquist plot of a passive system is in $Re(s) \geq 0$.

The complete master/slave system is illustrated in Figure 3. For an ideal communication channel

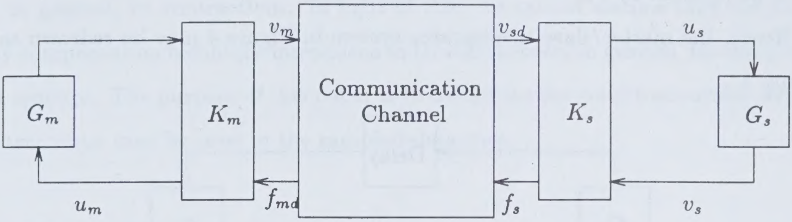


Figure 3: Master/Slave System

$$\begin{aligned} v_{sd}(t) &= v_m(t) \\ f_{md}(t) &= f_s(t) \end{aligned}$$

When a communication delay (of T seconds) exists, the communication channel equations become

$$\begin{aligned} v_{sd}(t) &= v_m(t-T) \\ f_{md}(t) &= f_s(t-T) \end{aligned}$$

which may lead to an unstable system.

We will analyze the compensation technique presented in [1]. The control law for the communication channel in [1] is given by:

$$\begin{aligned} v_{sd}(t) &= v_m(t-T) - f_s(t) + f_{md}(t-T) \\ f_{md}(t) &= f_s(t-T) - v_{sd}(t-T) + v_m(t) \end{aligned}$$

This results in a master/slave system as illustrated in Figure 4. In this figure, let

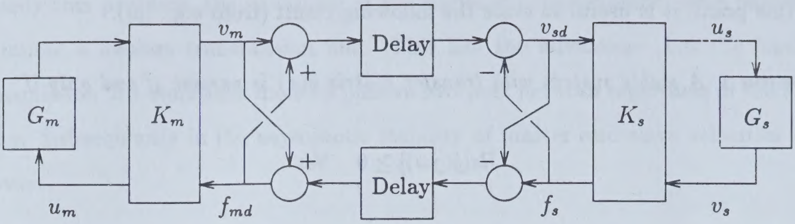


Figure 4: Master/Slave System

the operator from v_{sd} to f_s be denoted by F_s. Similarly, let the operator from f_{md} to v_m be denoted by F_m.

Hence, the master/slave teleoperator system in Figure 4 may be redrawn as in Figure 5

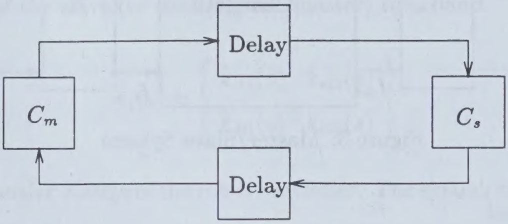


Figure 5: Master/Slave System

where

$$C_s = (F_s - I)(F_s + I)^{-1} \tag{1}$$

and

$$C_m = -(F_m + I)(F_m - I)^{-1} \tag{2}$$

Both $-F_m$ and F_s are assumed to be passive operators (i.e., $Re(-\hat{f}_m(j\omega) \geq 0$ and $Re(\hat{f}_s(j\omega) \geq 0)$. This is a reasonable assumption since both G_m and G_s are passive and the controllers K_m and K_s may be made up of passive elements. The negative sign on the master system is due to the fact that the feedback force, f_{md}, is opposite in direction to the commanded velocity, v_m. The operators C_s and C_m are contractions since (equations (1) and (2)) they're simply the bilinear transforms of $-F_m$ and F_s. The time delay element in the loop adds only phase to the system.

Thus, the system of Figure 5 is guaranteed to be stable for any time delay.

In practice, controllers for both the master and slave networks will generally be implemented digitally via a computer and the transmission of data would preferably occur over a digital channel. We note that if G is a passive strictly causal continuous-time system, its discrete-time counterpart (via ideal sampler and zero-order hold) is *not* passive for any sampling period, h. In fact, it can be shown that all strictly causal discrete-time systems are not passive (one proof uses the Discrete Positive Real Lemma [7] and [8]). Thus, the discrete-time counterparts of C_m and C_s will not, in general, be contractions. In light of this, we cannot assume that the time delay compensation technique introduced in [1] will succeed, in general, for sampled-data systems. The purpose of this paper is to determine the conditions under which this technique may be used in the sampled-data case.

3 Main Result

Let $\mathcal{C}$ denote the space of continuous functions $\mathbf{R} \to \mathbf{C}^n$, and $\mathcal{PC}$ the space of piecewise continuous functions $\mathbf{R} \to \mathbf{C}^n$. Also let $\mathbf{s}(\mathbf{C}^n)$ denote the space of sequences $\{x_0, x_1, \cdots\}$, $x_k \in \mathbf{C}^n$. We will suppress the $\mathbf{C}^n$ and denote this space simply by $\mathbf{s}$.

Denote by h the sampling period of a sampled-data system. Let S represent the sampling operator given by

$$S \; : \; \mathcal{C} \to \mathbf{s} \qquad y = Su$$
$$y(k) = u(kh)$$

and let H represent the zero-order hold operator given by

$$H \; : \; \mathbf{s} \to \mathcal{PC} \qquad u = Hu$$
$$y(t) = u(k), \quad (kh \leq t < (k+1)h)$$

We will consider only the slave side of the master/slave system (the master side follows identically). Figure 6 illustrates the sampled-data version of the slave system. Note that the map from the signal v_{sd} to f_s is in general not a passive one.

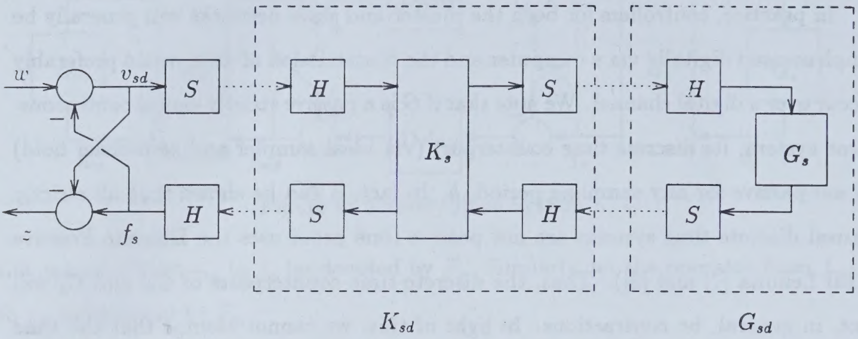


Figure 6: Sampled-data Slave System

This implies that the map from w to z is in general not a contraction. Thus, we cannot guarantee stability of the master/slave system when there is a time-delay in the communication channel.

In order to ensure stability for the sampled-data system when there is a time delay, we introduce three stable, lti, strictly causal filters: W_{1s}, W_{2s}, and W_{3s}. Consider the system in Figure 7. Generally, W_{is}, $i = 1, \ldots, 3$ are selected to have

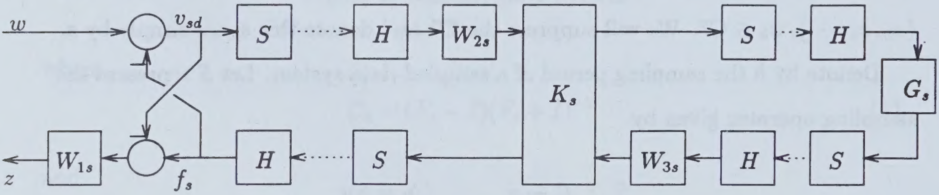


Figure 7: Sampled-data Slave System

unity gain at low (i.e., operating) frequencies and roll off at high frequencies. Hence, in the operating frequency band (e.g., up to 10 Hz for some robots), the systems of Figure 6 and Figure 7 behave identically.

Theorem 1 *For the sampled-data system of Figure 7,* $\exists \bar{h} > 0$ *such that the mapping from* w *to* z *is a contraction* $\forall$ *sampling periods* $h \leq \bar{h}$

In order to implement the controller with the filters W_{2s} and W_{3s}, we introduce the discrete operator

$$K_{wsd} = \begin{bmatrix} SK_{s11}W_{2s}H & SK_{s12}W_{3s}H \\ SK_{s21}W_{2s}H & SK_{s22}W_{3s}H \end{bmatrix} = \begin{bmatrix} K_{ws11d} & K_{ws12d} \\ K_{ws21d} & K_{ws22d} \end{bmatrix} \tag{3}$$

Corresponding to this operator is the discrete-time transfer function

$$\hat{k}_{wsd}(z) = \begin{bmatrix} \hat{k}_{ws11d}(z) & \hat{k}_{ws12d}(z) \\ \hat{k}_{ws21d}(z) & \hat{k}_{ws22d}(z) \end{bmatrix}$$

Similarly, for the master system, we introduce stable, lti, strictly causal filters: W_{1m}, W_{2m}, and W_{3m} and form the operator K_{wmd} as in equation (3) with transfer function $\hat{k}_{wmd}(z)$. The complete master/slave sampled-data teleoperator system is illustrated in Figure 8

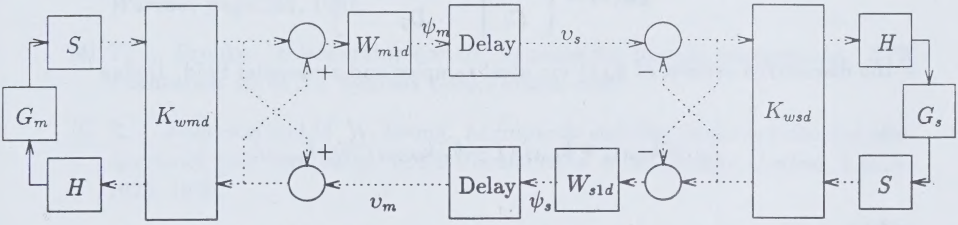


Figure 8: Master/Slave Sampled-data System

where

$$W_{s1d} = SW_{s1}H$$

and

$$W_{m1d} = SW_{m1}H$$

Theorem 1 states that if the sampling period is sufficiently small, the system of Figure 8 is stable for any time delay. In order to determine a suitably small sampling period, we may use the following procedure:

1 choose an $h > 0$

2 compute the induced l_2 to l_2 norm of the slave system (right side of Figure 8)

3 if the norm is greater than one, go back to Step 1 and repeat with a smaller h (e.g., divide h by 2).

4 repeat procedure for the master system (left side of Figure 8)

Step 2 of the above procedure may be performed in the following way. First bring in a minimal realization for $\hat{g}_s(s)$:

$$\hat{g}_s(s) = \left[\begin{array}{c|c} A_s & B_s \\ \hline C_s & D_s \end{array}\right]$$

Then

$$\hat{g}_{sd}(z) = \left[\begin{array}{c|c} e^{A_s h} & \int_0^h e^{A_s \tau} d\tau B_s \\ \hline C_s & D_s \end{array}\right]$$

is the discretized version of $\hat{g}_s(s)$ via ideal sampler and zero-order hold. Define

$$\hat{f}_{sd} = \hat{k}_{ws11d} + \hat{k}_{ws12d}(I - \hat{g}_{sd}\hat{k}_{ws22d})^{-1}\hat{g}_{sd}\hat{k}_{ws21d}$$

and let

$$\hat{c}_{sd} = (\hat{f}_{sd} - I)(\hat{f}_{sd} + I)^{-1}$$

Then,

$$\sup_{\|v_s\|_2} \{\|\psi_s\|_2\} = \sup_{\theta\in[0,2\pi)} \sigma_{max}\left\{\hat{c}_{sd}(e^{j\theta})\hat{w}_{s1d}(e^{j\theta})\right\}$$

where $\sigma_{max}(\cdot)$ is the maximum singular value.

4 Conclusions

In this paper, we have presented a method for the bilateral control of a teleoperator system through a digital communication channel in the presence of a time delay.

This method is based on the analog control method in [1]. For passive master and slave systems, G_m and G_s, with passive stabilizing controllers, K_m and K_s, a procedure has been given to determine a suitable sampling period h which ensures stability of the sampled-data master/slave system for any time delay in the communication channel.

References

[1] R. J. Anderson and M. W. Spong. Bilateral control of teleoperators with time delay. In *IEEE Conference on Decision and Control*, pages 167–173, 1988.

[2] W. R. Ferrell. Remote manipulation with transmission delay. *IEEE Transactions on Human Factors in Electronics*, HFE-6:24–32, September 1965.

[3] J. Vertut, A. Micaelli, P. Marchal, and J. Guittet. Short transmission delay on a force relective bilateral manipulator. In *Proceedings of the 4th Rom-An-Sy, Warsaw*, page 269, 1981.

[4] T. L. Brooks. Teleoperator system response for nuclear telepresence. *STX Publication 90-03 ST Systems Corp.*, March 1990.

[5] R. J. Anderson and M. W. Spong. Asymptotic stability for force reflecting teleoperators with time delay. *IEEE Transactions on Automatic Control*, 8:1618–1625, 1989.

[6] C. A. Desoer and M. Vidyasagar. *Feedback Systems: Input-Output Properties.* Electrical Science Series, Academic Press, 1975.

[7] B. D. O. Anderson and S. Vongpanitlerd. *Network Analysis and Synthesis: A Modern Systems Theory Approach.* Prentice Hall, 1973.

[8] L. Hitz and B. D. O. Anderson. Discrete positive-real functions and their application to system stability. In *Proceedings of the IEEE Conference on Automatic Control*, volume 116, pages 153–155, 1969.

[9] T. Chen. *Control of Sampled-Data Systems.* Ph.D. Thesis, Dept. of Electrical Engineering, University of Toronto, 1991.

[10] G. M. H. Leung, T. P. Perry, and B. A. Francis. Performance analysis of sampled-data systems. *Automatica*, 27(4):699–705, 1991.

Hybrid Position/Force Control of Redundant Robot with Consideration of Motor Dynamics*

T.J. Tarn, A.K. Bejczy** and Ning Xi

Dept. of Systems Science and Mathematics
Washington University
Saint Louis, MO 63130 – 4899

**Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA91109

Abstract

This paper presents a new hybrid position/force control scheme for redundant manipulator. The joint motor dynamics is taken into account, which plays an important role in the force control. The concept of general task space is introduced. Combining with well known nonlinear feedback technique, the robot arm and joint motor dynamic models are linearized and decoupled in the general task space. Based on a given task, a task projection operator may be found with the consideration of redundancy management. It projects the linearized model to actual task space. This scheme were experimentally implemented and tested on a 6-DOF position/force control of a PUMA 560 robot arm with very good results.

1 Introduction

It is essential to control the motion of a robot and the output force of the robot simultaneously in order to perform some sophisticated tasks such as assembly, polishing, deburring, etc. Since force control is highly dependent on the task that the robot performs, most position/force hybrid controllers were designed only for some specific tasks. One of the important characteristics of an intelligent robotic system is its ability to work in a nonstructured environment. In other words, it should be able to perform different tasks without redesigning the control system.

It was shown that the motor dynamics played a very important role in the position control of a robot manipulator [1]. In order to achieve both good transient response and small steady state tracking errors, it is necessary to include the motor dynamics in the hybrid position/force control model.

The linearizing and decoupling approach to simultaneous position and force control has been in vogue [2–5]. In [6–8], a dynamic hybrid position/force controller for a robot manipulator is formulated.

In this paper a general task space is introduced. Based on a well known nonlinear feedback scheme, the manipulator and the motor dynamic models are linearized and decoupled with respect to outputs defined in the general task space. Then a task projection operator is introduced, which projects the general output to a controllable subspace of general task space, i.e. actual task space. Separate linear controllers are designed to control the position, force and redundant joints.

The ultimate goal of this paper is to develop a hybrid position/force controller which considers the joint motor dynamics. More importantly it is desirable that the same controller be used in the performance of different tasks. In addition, the redundant input of the system can easily be resolved to achieve secondary goals. This approach yields a systematic way to manage redundancy of robotic system.

---

*This work was supported in part by the National Science Foundation under Grant IRI-9106317 and by the Sandia National Laboratories under Contract AC-3752-C.

2 Effect of Motor Dynamics on Force Control

Without considering the joint motor dynamics, the nonredundant robot dynamics can be modeled as

$$D(q)\ddot{q} + C(q,\dot{q}) + G(q) + J^T(q)f = \tau \tag{2.1}$$

where

q : Joint angle vector
$D(q)$: Inertia matrix
$C(q,\dot{q})$: Centripetal and Coriolis terms
$G(q)$: Gravity loading
τ : Joint torque vector
$J(q)$: Jacobian matrix, such that $\dot{y} = J(q)\dot{q}$
y : Position and orientation in task space.

Applying the well known nonlinear feedback control law

$$\tau = D(q)J^{-1}(q)\left[V_1 - \dot{J}(q)\dot{q}\right] + C(q,\dot{q}) + G(q) + J^T(q)V_2, \tag{2.2}$$

if $y \perp f$, which implies that the force and position are not commanded in the same axis, the linearized model is

$$\begin{cases} \ddot{y} = V_1 \\ f = V_2 \end{cases} \tag{2.3}$$

where

V_1 : Position and orientation command
V_2 : Force command.

It is evident that there is no dynamics between the input command and the output force in (2.3). The command affects the output instantaneously.

The commonly used force control law is

$$V_2 = f^d + K_{fp}\left(f^d - f\right) + K_{fi}\int_0^t \left(f^d - f\right) dt. \tag{2.4}$$

where f^d is the desired force. K_{fp}, K_{fi} are proportional and integral feedback gains.

It can be easily shown that based on the linearized model $f = V_2$, the feedback law

$$V_2 = f^d + K_{fp}\left(f^d - f\right) + K_{fi}\int_0^t \left(f^d - f\right) dt$$

is equivalent to

$$V_2 = f^d + \frac{K_{fi}}{1 + K_{fp}}\int_0^t \left(f^d - f\right) dt. \tag{2.5}$$

Therefore, the proportional control actually plays no role in the force feedback control law. As a result, it would be very difficult to achieve both small steady state tracking error and quick transient response. Furthermore, if there is no integral feedback, i.e. $K_{fi} = 0$, the force control turns into an open loop system. Since there is no dynamics between the input and output of the force control loop, the motor dynamics becomes the dominant part. It, hence has to be included in the dynamic model in order to achieve our goals.

3 Hybrid Force/Position Controller with the Consideration of Motor Dynamics

The n-joint manipulator and motor dynamic model can be written as

$$\begin{cases} D(q)\ddot{q} + C(q,\dot{q}) + G(q) + \tau_f = \tau \\ \dfrac{d\tau}{dt} = -T^{-1}\tau + K_e\dot{q} + u \end{cases} \tag{3.1}$$

where

τ_f : Joint torque which produces output force f.
T : $n \times n$ diagonal matrix, T_i is time constant of ith motor.
K_e : Voltage constant of the motor (back EMF).
u : Motor armature voltage.

and $q, \tau, \in \mathrm{I\!R}^n$.

The general task space is defined as

$$\mathcal{T} = \mathcal{Y} \bigoplus \mathcal{F} \bigoplus \mathcal{Z}$$

where

$\mathcal{Y} \in \mathrm{I\!R}^6$ is the general position space.
$\mathcal{F} \in \mathrm{I\!R}^6$ is the general force space.
$\mathcal{Z} \in \mathrm{I\!R}^n$ is the general redundant joint space.

$$\mathcal{T} = \left\{ \begin{bmatrix} y \\ f \\ q \end{bmatrix} \in \mathrm{I\!R}^{12+n} \right\}$$

Since there are only n joints available, the manipulator can only be controlled in a subspace of $\mathcal{T}$, $\mathcal{T}_c \in \mathrm{I\!R}^n$. The actual output, hence is in $\mathcal{T}_c$.

$\mathcal{T}_c$ is defined as

$$\mathcal{T}_c = \mathcal{Y}_c \bigoplus \mathcal{F}_c \bigoplus \mathcal{Z}_c$$

where

$$\mathcal{Y}_c \in \mathrm{I\!R}^{k_1}$$
$$\mathcal{F}_c \in \mathrm{I\!R}^{k_2}$$
$$\mathcal{Z}_c \in \mathrm{I\!R}^{n-(k_1+k_2)}.$$

The task projection operator $B_c : \mathcal{T} \longmapsto \mathcal{T}_c$ is the basis of $\mathcal{T}_c$ and is described as

$$B_c = [Y_c \; F_c \; Z_c].$$

The joint space torque that produces the output force f can be written as [2]

$$f = \left(J^T(q)\right)^{\#} \tau_f$$

where "#" denotes a pseudo inverse.

Because of redundancy, the joint torque is not unique for a given f, it can be stated as [2]

$$\tau_f = J^T(q)f + \left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\Gamma_f$$

where I_n is $n \times n$ identity matrix,

$$\left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]$$

is the null space of $\left(J^T(q)\right)^{\#}$ and Γ_f is any $n \times 1$ vector.

By a similar argument,

$$\ddot{q} = J^{\#}(q)\left(\ddot{y} - \dot{J}(q)\dot{q}\right) + \left[I_n - J^{\#}(q)J(q)\right]\Gamma_y$$

where $\left[I_n - J^{\#}(q)J(q)\right]$ is the null space of $J(q)$ and Γ_y is any $n \times 1$ vector.

Hence, (3.1) can be written as

$$\begin{aligned} D(q)J^{\#}(q)\left(\ddot{y} - \dot{J}(q)\dot{q}\right) + D(q)[I_n - J^{\#}(q)J(q)]\Gamma_y + C(q,\dot{q}) + G(q) \\ + J^T(q)f + \left[I_n - J^T(q)(J^T(q))^{\#}\right]\Gamma_f = \tau \end{aligned}$$

If choose $\Gamma_y = 0$ and $\Gamma_f = \ddot{q}$, then

$$\left\{\begin{aligned} & D(q)J^{\#}(q)\left(\ddot{y} - \dot{J}(q)\dot{q}\right) + C(q,\dot{q}) + G(q) \\ & \qquad + J^T(q)f + \left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\ddot{q} = \tau \\ & \frac{d\tau}{dt} = -T^{-1}\tau + K_e\dot{q} + u \end{aligned}\right. \tag{3.2}$$

let

$$\tau_1 = D(q)J^{\#}(q)(\ddot{y} - \dot{J}(q)\dot{q}) + C(q,\dot{q}) + G(q)$$

$$\tau_2 = J^T(q)f$$

$$\tau_3 = \left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\ddot{q}$$

then $\tau = \tau_1 + \tau_2 + \tau_3$.

In addition, let

$$u_1 = \frac{d\tau_1}{dt} + T^{-1}\tau_1 - K_e\dot{q}$$

$$u_2 = \frac{d\tau_2}{dt} + T^{-1}\tau_2$$

$$u_3 = \frac{d\tau_3}{dt} + T^{-1}\tau_3$$

then $u = u_1 + u_2 + u_3$.

As a result of the above definitions, the dynamic model (3.2) has been separated in general task space $\mathcal{T}$.

In general position space $\mathcal{Y}$:

$$\left\{\begin{aligned} & D(q)J^{\#}(q)\left(\ddot{y} - \dot{J}(q)\dot{q}\right) + C(q,\dot{q}) + G(q) = \tau_1 \\ & \frac{d\tau_1}{dt} = -T^{-1}\tau_1 + K_e\dot{q} + u_1 \end{aligned}\right. \tag{3.3}$$

In general force space $\mathcal{F}$:

$$\left\{\begin{aligned} & J^T(q)f = \tau_2 \\ & \frac{d\tau_2}{dt} = -T^{-1}\tau_2 + u_2 \end{aligned}\right. \tag{3.4}$$

In general redundant joint space $\mathcal{Z}$:

$$\begin{cases} \left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\ddot{q} = \tau_3 \\ \dfrac{d\tau_3}{dt} = -T^{-1}\tau_3 + u_3 \end{cases} \tag{3.5}$$

With consideration of the motor dynamics, acceleration measurements are necessary for linearization and decoupling in general position and redundant joint spaces [1]. Practically, it is very difficult to get the acceleration information. Therefore, the motor dynamics will only be considered in general force space.

After ignoring the motor dynamics in general position and redundant joint spaces,(3.3) and (3.5) could be rewritten as

$$D(q)J^{\#}(q)\left(\ddot{y} - \dot{J}(q)\dot{q}\right) + C(q,\dot{q}) + G(q) = TK_e\dot{q} + Tu_1, \tag{3.6}$$

and

$$\left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\ddot{q} = Tu_3 \tag{3.7}$$

Based on well known nonlinear feedback technique, the nonlinear controls,

$$\begin{aligned} u_1 &= T^{-1}\left(D(q)J^{\#}(q)V_1 - D(q)J^{\#}(q)\dot{J}(q)\dot{q}\right. \\ &\qquad \left. + C(q,\dot{q}) + G(q) - TK_e\dot{q}\right) \\ u_2 &= \left(T^{-1}J^T(q) + \dot{J}^T(q)\right)f + J^T(q)V_2 \\ u_3 &= T^{-1}\left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]V_3 \end{aligned}$$

can linearize and decouple the position, force and redundant joint dynamic models in $\mathcal{Y}$, $\mathcal{F}$ and $\mathcal{Z}$. The linearized models are

$$\begin{aligned} \ddot{y} &= V_1 \\ \dot{f} &= V_2 \\ \ddot{q} &= V_3 \end{aligned}$$

where $V_1, V_2, \in \mathbb{R}^6$ and $V_3 \in \mathbb{R}^n$ are auxiliary inputs.

However, since there are only n controls available, the above linearization and decoupling are not actually feasible in $\mathcal{T}$. But if we only consider the controllable subspace $\mathcal{T}_c$, the linearization and decoupling can be achieved.

It can be proven that if the position and force are not controlled in the same axis in task space, then the nonlinear feedback control,

$$\begin{aligned} u = T^{-1}&\left(D(q)J^{\#}(q)\bar{V}_1 - D(q)J^{\#}(q)\dot{J}(q)\dot{q} + C(q,\dot{q}) + G(q) - TK_e\dot{q}\right) \\ &-T^{-1}J^T(q)\bar{f} + J^T(q)\bar{V}_2 + T^{-1}\left[I_n - J^T(q)\left(J^T(q)\right)^{\#}\right]\bar{V}_3 \end{aligned} \tag{3.8}$$

linearizes and decouples the dynamic system (3.2) in $\mathcal{T}_c$. The linearized and decoupled system is

$$\begin{aligned} \ddot{\bar{y}} &= \bar{V}_1 \\ \dot{\bar{f}} &= \bar{V}_2 \\ \ddot{\bar{q}} &= \bar{V}_3 \end{aligned} \tag{3.9}$$

where

$$\begin{bmatrix} \bar{y} \\ \bar{f} \\ \bar{q} \end{bmatrix} = B_c^T \begin{bmatrix} y \\ f \\ q \end{bmatrix} \text{ and } \begin{bmatrix} \bar{V}_1 \\ \bar{V}_2 \\ \bar{V}_3 \end{bmatrix} = B_c^T \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}.$$

Then the linear controllers

$$\begin{aligned} \bar{V}_1 &= \ddot{\bar{y}}^d + k_v \left(\dot{\bar{y}}^d - \dot{\bar{y}}\right) + k_p \left(\bar{y}^d - \bar{y}\right) \\ \bar{V}_2 &= \dot{\bar{f}}^d + k_f \left(\bar{f}^d - \bar{f}\right) \\ \bar{V}_3 &= \ddot{\bar{q}}^d + k_{\dot{q}} \left(\dot{\bar{q}}^d - \dot{\bar{q}}\right) + k_q \left(\bar{q}^d - \bar{q}\right) \end{aligned} \tag{3.10}$$

can be used to stabilize and control position, force and redundant joints.

Remarks:

- It can be shown that for different tasks, the structures of controllers (3.8) are the same. Only changes must be made in the task projection operator B_c. Therefore, the task definition with a hybrid position/force control becomes designing a suitable B_c. In this respect, the controller structure is task-independent.

- The redundancy of the system is described by the null space of $\left(J^T(q)\right)^{\#}$. The management of redundancy consists of designing B_c to achieve certain goals. In addition, once the redundant joints are determined, their motion can be planned and controlled based on some secondary optimization criteria: such as obstacle avoidance, load sharing etc.

- By considering the motor dynamics, the force loop dynamic model (3.4) becomes nonlinear. Obviously, it would not be suitable to use the traditional linear controller (2.4) to control such a nonlinear system. It is necessary to apply a nonlinear feedback control (3.8) to linearize and decouple the system. Then a proper linear control law (3.10) can be used.

4 Experimental Implementation

The above control algorithm was implemented and tested for hybrid position/force control on a 6-DOF PUMA 560 robot arm which was equipped with a FSA-3254 6-Axis Force/Torque Sensor. The force/torque was measured at a rate of 1000 Hz, and the filitered force/torque was computed at a rate of 500 Hz. The sampling rate for control and joint position/velocity measurement was 1000 Hz. In the experiments the positions in x and y directions and orientation (O,A,T) were commanded. In z direction, only force was commanded. Two different controllers, one with the consideration of joint motor dynamics and the other without the consideration of the joint motor dynamics, were used to perform various tasks. In Figure 1, the constant desired force was tracked. It can be seen that the controller with the joint motor dynamics gave a better force tracking. In addition, both controllers maintained good x and y position tracking. Since the z position was not commanded, the error in z direction was much bigger than x and y.

In Figures 2 and 3, results for tracking variant desired forces were given. The advantage of the controller with the joint motor dynamics was more obviously shown in these cases. The control law (2.4) was used in the controller without the joint motor dynamics. Obviously, it took some time to establish the feedback from the integrate term. As a result, it performed poorly when trakcing a variant input.

The time-constant of the motors for the first three joints of PUMA 560 robot is 3ms, and that for the last three joints is 1ms. Therefore, if the higher sampling rate were applied in force measurement, better results would be expected.

5 Conclusion

A hybrid position/force controller is proposed in this paper that incorporates the robot manipulator dynamics as well as the dynamics of robot joint motors. The concept of general task space is introduced. The robot and motor dynamic models are linearized and decoupled in this general task space. Based on a given task, a task projection operator can be found with consideration of redundancy management. The linearized model can be projected by the projection operator to the actual task space. As a result, the controller structure is independent of individual task. Due to the inclusion of the motor dynamics, the force control performance is significantly improved, especially in the case of tracking variant desired force. More importantly, the structure of the controller is independent of task. This makes it possible for a robot to work in a nonstructured environment. It could also be an important step to the development of an intelligent robot system.

References

[1] T.J. Tarn, A.K. Bejczy, X. Yun, Z. Li, "Effect of Motor Dynamics on Nonlinear Feedback Robot Arm Control," IEEE Transactions on Robotics and Automation, Vol. 7, No. 1, Feb. 1991.

[2] O. Khatib, "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation," IEEE Journal of Robotics and Automation, Vol. RA-3, No. 1, Feb. 1987.

[3] T. Yoshikawa, "Dynamic Hybrid Position/Force Control of Robot Manipulators – Description of Hand Constraints and Calculation of Joint Driving Force," IEEE Journal of Robotics and Automation, Vol. RA-3, No. 5, Oct. 1987.

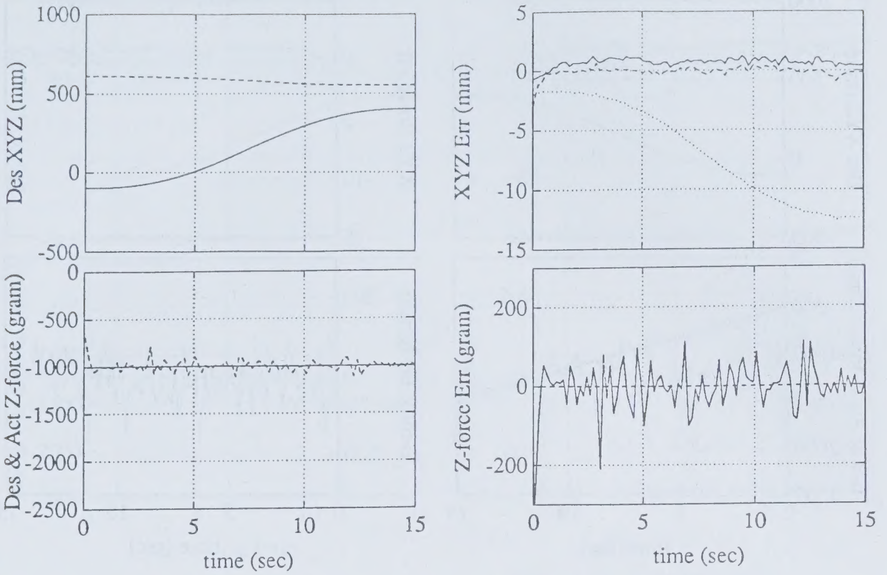
[4] T.J. Tarn, A.K. Bejczy, X. Yun, "Robot Arm Force Control through System Linearization by Nonlinear Feedback," Proc. of IEEE Conf. on R&A, Philadelphia, 1988.

[5] A.K. Ramadorai, T.J. Tarn, A.K. Bejczy, "Task Definition, Decoupling and Redundancy Resolution by Nonlinear Feedback in Multi-Robot Object Handling," Proc. of IEEE Conf. on R& A, Nice, France, May 1992.

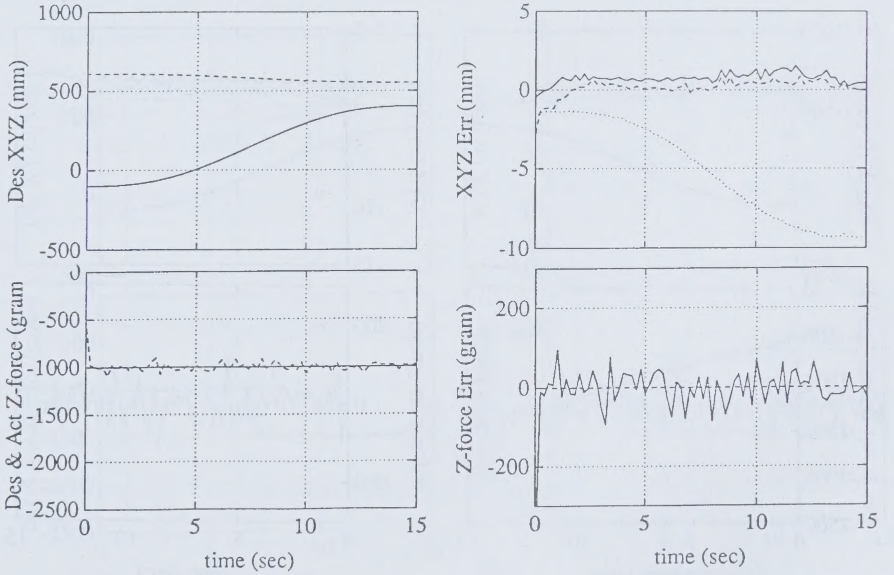
[6] M.H. Raibert, J.J. Craig, "Hybrid Position/Force Control of Manipulators," Trans. of ASME, Vol. 102, June 1981.

[7] M.T. Mason, "Compliandce and Force Control of Computer Controlled Manipulators," IEEE Trans. on Systems, Man and Cybernetics, Vol. SM C-11, No. 6, June 1981.

[8] S. Hayati, "Hybrid Positon/Force Control of Multi-Arm Cooperating Robots," Proc. of IEEE Conf. on R& A, San Francisco, 1986.

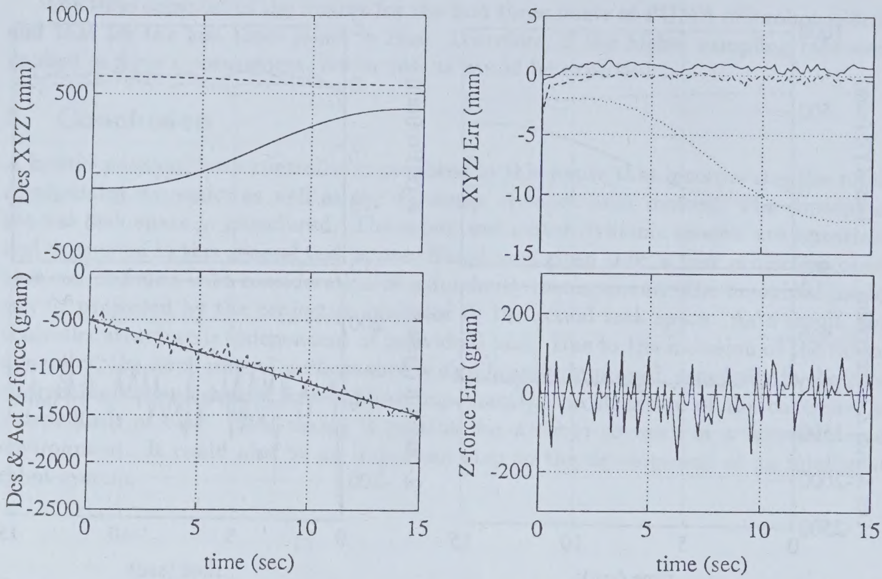


Without the Consideration of Joint Motor Dynamics

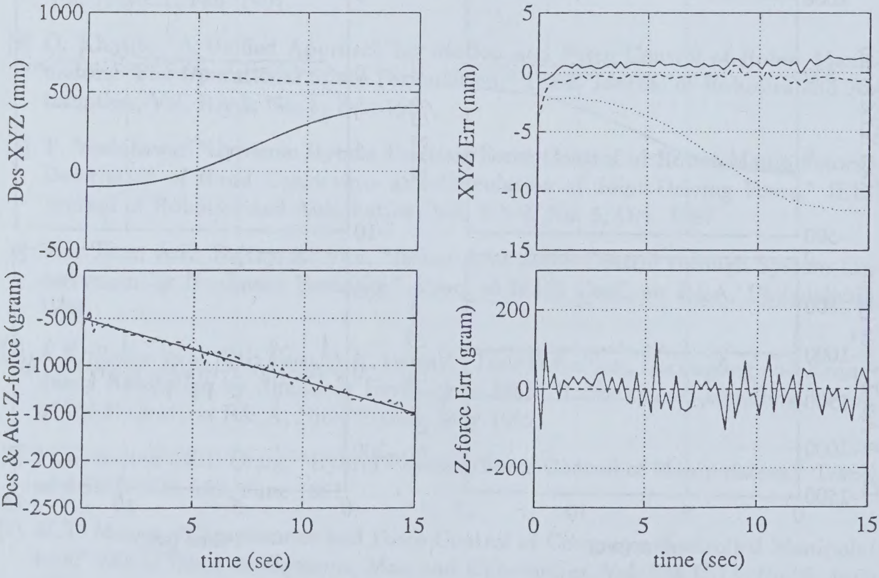


With the Consideration of Joint Motor Dynamics

Figure 1. Tracking Constant Force

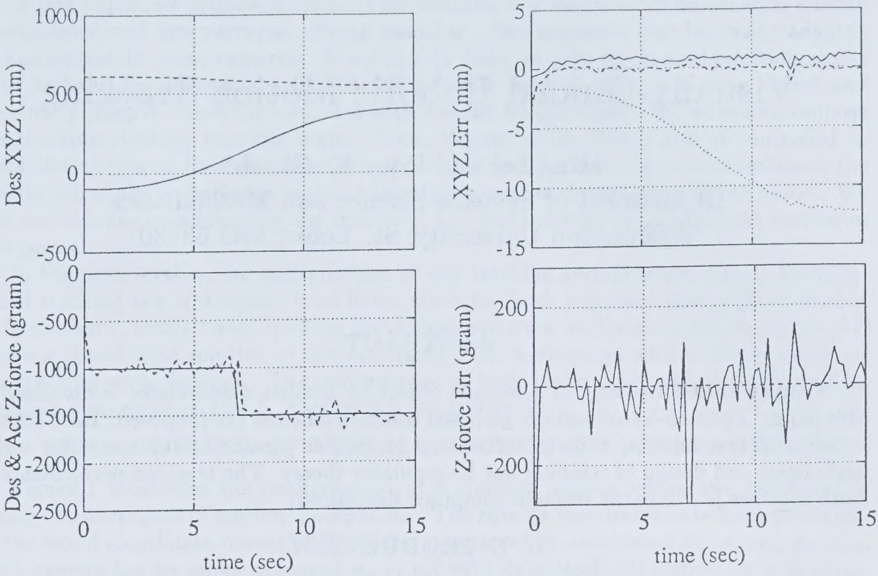


Without the Consideration of Joint Motor Dynamics

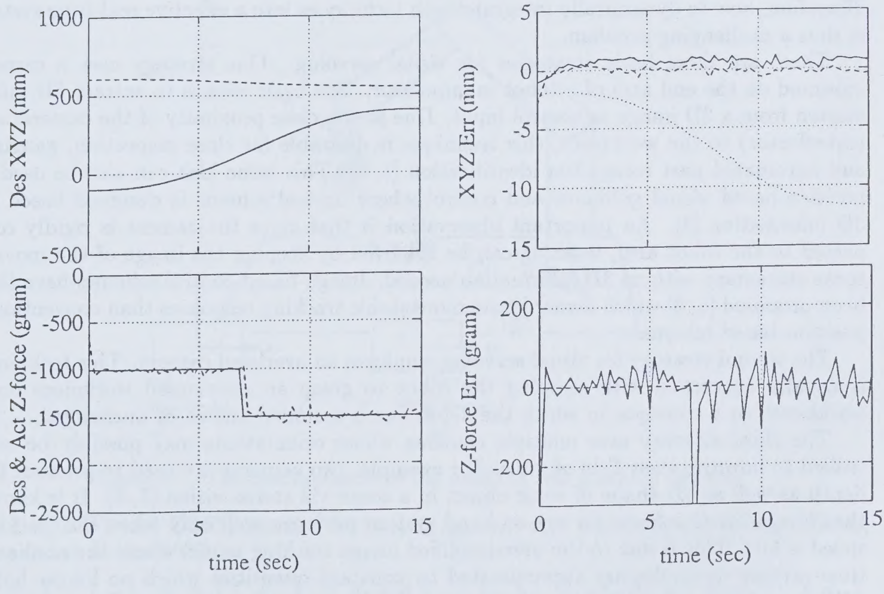


With the Consideration of Joint Motor Dynamics

Figure 2. Tracking Slope Force



Without the Consideration of Joint Motor Dynamics



With the Consideration of Joint Motor Dynamics

Figere 3. Tracking Step Force

Visually-Guided Robotic Motion Tracking

Ming Lei and Bijoy K. Ghosh*
Department of Systems Science and Mathematics
Washington University, St. Louis, MO 63130

ABSTRACT

Visually-guided tracking of a moving object by a robot manipulator is discussed in this paper. Position-based and image-based tracking schemes are proposed. Each scheme consists of two aspects, namely, estimation of motion parameters by computer vision techniques and design of tracking law by regulator theory. The tracking performance of both schemes is compared through computer simulation.

1. INTRODUCTION

Visually-guided robot control(visual servo) is important in dealing with the problems involving unknown environments and uncertain models. This problem contains two aspects–robot control and computer vision. The knowledge of the scene is accquired from processing the images, and is sent to robot controller to maneuver the robot accordingly. Therefore, how to dynamically integrate both techniques into a effective real-time system is thus a challenging problem.

There are three basic strategies for visual servoing. One strategy uses a camera mounted on the end arm of a robot manipulator. The basic idea is to extract 3D information from a 2D image as control input. Due to the close proximity of the camera(and end-effector) to the workpiece, this technique is desirable for close inspection, gauging, and automated part recogniton identification [1, 2]. This same idea can also be used in position-based visual guidance and control where control scheme is designed based on 3D information [3]. An important observation is that since the camera is rigidly connected to the robot arm, tracking can be achieved by keeping the image of the moving scene stationary with no 3D information needed. Image-based control schemes have thus been proposed [4, 5] which demonstrate more stable tracking responses than conventional position-based schemes.

The second strategy for visual servoing employes an overhead camera. This technique is usually concerned with servoing the robot to grasp an unoriented workpiece on a workbench or a conveyor in which the depth cue is assumed known or unchanged.

The third strategy uses multiple cameras whose orientations may possibly be controlled to improve their field of view. For example, two cameras are used to produce the depth as well as 3D shape of some object in a scene via stereo vision [7, 8]. It is known that image-based scheme for eye-on-hand system performs well only when the tracking speed is low. This is due to the oversimplified image tracking model where the nonlinear time-varying dynamics are approximated by constant quantities which no longer holds when speed increases. On the other hand, with multiple cameras, more information is obtained and the dynamics of the visual servoing are better modeled.

---

*Partially supported by DOE under grant No. DE-FG02-90ER14140.

In this paper, we propose a modular formulation of a visual servo model with a robot manipulator and two cameras. Being modular , the approach can be easily adapted to accommodate more cameras. A six-link (3 links as arm for spatial position and 3 links as hand for orientation) PUMA 560 robot is used to track a moving object and eventually grasp it. Since the control of each link can be decoupled [10], we can decompose the tracking strategy into two steps. First, the tip of the 3-link arm is controlled to track the motion of the centroid of the object. When the tracking is accomplished, the orientation of the end-effector is then controlled to insure the grasping of the object. We will concentrate on addressing the first step, i.e., the tracking of the centroid motion in this paper.

In the next section, the configuration of our tracking system is described. Position-based tracking law is designed from linear state feedback regulator theory [9] in section 3. In section 4, image-based tracking law is designed from nonlinear regulator theory [11] where a closed form solution of the associated PDE is obtained with a proper choice of the tracking error function. The performance of both tracking schemes is compared by simulation study in section 5. Section 6 is the conclusion.

2. THE MODELING OF VISUALLY-GUIDED TRACKING SYSTEM

Figure 1 illustrates the configuration of a tracking system where two cameras are facing the workspace of a robot manipulator. The robot's base coordinate frame is chosen as the world coordinate frame(WCF) of the system, which is denoted by $(o_w; x_w, y_w, z_w)$. Each camera has its center of lens at o_{c1} or o_{c2} with its optical axis coinciding with $o_{c1}z_{c1}$ or $o_{c2}z_{c2}$ and its image plane being located between the center of lens and the workspace of the robot. The distance from the center of lens to the image plane is called the focal length of the camera. The camera coordinate frame(CCF) is related to the WCF by some known rotation and translation.

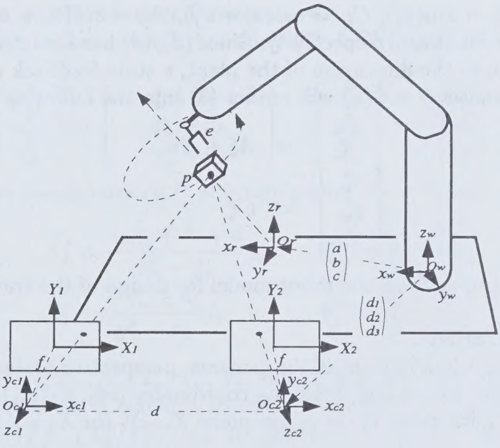


Figure 1 The configuration of the tracking and grasping system

Let CCF1 and CCF2, denoted by $(o_{c1}; x_{c1}, y_{c1}, z_{c1})$ and $(o_{c2}; x_{c2}, y_{c2}, z_{c2})$ respectively, be related to the WCF by

$$\begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{c1} \\ y_{c1} \\ z_{c1} \end{bmatrix} + \begin{bmatrix} d_1 + d \\ d_2 \\ d_3 \end{bmatrix}, \quad \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{c2} \\ y_{c2} \\ z_{c2} \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{1}$$

where d, d_1, d_2 and d_3 are known translational components.

For any point P in the workspace, its coordinates in the WCF, CCF1 and CCF2 are denoted respectively, by (x_w^p, y_w^p, z_w^p), $(x_{c1}^p, y_{c1}^p, z_{c1}^p)$ and $(x_{c2}^p, y_{c2}^p, z_{c2}^p)$.

2.1 The Model of the Robot Manipulator

For task of tracking the centroid motion of the object, we only consider the dynamics of the first three links of the PUMA 560 arm, which is described by [10]

$$D(q)\ddot{q} + P(q, \dot{q}) = u \tag{2}$$

where $q = [q_1\ q_2\ q_3]^T$, $u = [u_1\ u_2\ u_3]^T$. q_i, $\dot{q}_i$, $\ddot{q}_i$ $(i = 1, 2, 3)$ are position, velocity and acceleration of jonit i. u_i is the torque acting at joint i. By introducing state variables $x_1 = q_1$, $x_2 = q_2$, $x_3 = q_3$, $x_4 = \dot{q}_1$, $x_5 = \dot{q}_2$, $x_6 = \dot{q}_3$, (2) has the form of

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_4 \\ x_5 \\ x_6 \\ \\ D^{-1}(x)P(x) \\ \\ \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \\ D^{-1}(x) \\ \\ \end{bmatrix} u, \tag{3}$$

The position $(x_w^e\ y_w^e\ z_w^e)^T$ of the end-effector is

$$\begin{bmatrix} x_w^e \\ y_w^e \\ z_w^e \end{bmatrix} = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \end{bmatrix} \triangleq \begin{bmatrix} a_3C_1C_{23} + d_4C_1S_{23} + a_2C_1C_2 - d_3S_1 \\ a_3S_1C_{23} + d_4S_1S_{23} + a_2S_1C_2 + d_3C_1 \\ -a_3S_{23} + d_4C_{23} - a_2S_2 \end{bmatrix}, \tag{4}$$

where $C_i = cos(x_i)$, $S_i = sin(x_i)$, $C_{ij} = cos(x_i + x_j)$, $S_{ij} = sin(x_i + x_j)$. a_i and d_i are length and offset of the robot arm respectively. Since (3), (4) has a relative degree (2, 2, 2) whose components sum to the dimension of the plant, a state feedback $u = \alpha(x) + \beta(x)v$ and a change of coordinates $\xi = \Phi(x)$ will render (3) into the following form [11, 10]:

$$\dot{\xi} = A\xi + Bv \tag{5}$$

$$\begin{bmatrix} x_w^e \\ y_w^e \\ z_w^e \end{bmatrix} = C\xi \tag{6}$$

System (5), (6) will be used as the robot model for design of the tracking controllers.

2.2 The Model of Camera

Since the workspace is just in front of the cameras, perspective projection is chosen as our camera model. Thus, for a point P having coordinates $(x_{c1}^p, y_{c1}^p, z_{c1}^p)$(or $(x_{c2}^p, y_{c2}^p, z_{c2}^p)$) in the CCF1 (or CCF2), its image in the image plane $X_1 - Y_1$ (or $X_2 - Y_2$) has coordinates

$$X_1^p = \frac{x_{c1}^p}{z_{c1}^p} f,\ Y_1^p = \frac{y_{c1}^p}{z_{c1}^p} f\ \text{(or } X_2^p = \frac{x_{c2}^p}{z_{c2}^p} f,\ Y_2^p = \frac{y_{c2}^p}{z_{c2}^p} f\text{)}. \tag{7}$$

2.3 The Model of the Centroid Motion

In order to apply the regulator theory, the exosystem being tracked must be Poisson stable. In particular, an object moving circularly in the workspace of the robot is chosen as our motion model. Although the discussions in the sequel assume constant motion velocity, the resulting scheme can be easily adapted for motion with small velocity variation.

Suppose now the centroid of the object, denoted by P, is moving along some arbitrary circle in 3D space. A local coordinate frame for the rotation which we shall call rotation coordinate frame(RCF), denoted by $(o_r; x_r, y_r, z_r)$, can be set up (see Figure 1). The rotation of the centroid P is easily modeled by

$$\overset{\cdot}{\begin{bmatrix} x_r^p \\ y_r^p \\ z_r^p \end{bmatrix}} = \Omega \begin{bmatrix} x_r^p \\ y_r^p \\ z_r^p \end{bmatrix}, \quad \Omega = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}, \quad \begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix} = \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} - \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{8}$$

with ω_1, ω_2 and ω_3 being some unknown constant motion parameters. So P moves in the WCF according to

$$\overset{\cdot}{\begin{bmatrix} x_w^p \\ y_w^p \\ z_w^p \end{bmatrix}} = \Omega \begin{bmatrix} x_w^p \\ y_w^p \\ z_w^p \end{bmatrix} + \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}, \quad \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} = -\Omega \begin{bmatrix} a \\ b \\ c \end{bmatrix}. \tag{9}$$

In order to know the rotation, the 3D motion parameters ω_1, ω_2, ω_3, a', b', c', need to be estimated.

3. POSITION-BASED TRACKING

When the centroid P moves in the 3D space, it induces optical-flow in the image planes of camera c1 and c2. By measuring these optical flows at time instant t, we can find the 3D position and velocity of P at that moment. The 3D motion parameters in (9) are then recursively estimated. Finally,we design a tracking law by solving a state feedback regulator problem(SFRP).

3.1 Estimation of 3D Motion Parameters

The coordinates of the centroid P at the WCF, CCF1 and CCF2 have the relationship, by (1),

$$\begin{bmatrix} x_{c1}^p \\ y_{c1}^p \\ z_{c1}^p \end{bmatrix} = \begin{bmatrix} -x_w^p + d_1 + d \\ z_w^p - d_3 \\ y_w^p - d_2 \end{bmatrix}, \quad \begin{bmatrix} x_{c2}^p \\ y_{c2}^p \\ z_{c2}^p \end{bmatrix} = \begin{bmatrix} -x_w^p + d_1 \\ z_w^p - d_3 \\ y_w^p - d_2 \end{bmatrix}.$$

By (7), the image coordinates of P in image planes $X_1 - Y_1$ and $X_2 - Y_2$ are

$$X_1^p = \frac{-x_w^p + d_1 + d}{y_w^p - d_2} f, \qquad Y_1^p = \frac{z_w^p - d_3}{y_w^p - d_2} f; \tag{10}$$

$$X_2^p = \frac{-x_w^p + d_1}{y_w^p - d_2} f, \qquad Y_2^p = \frac{z_w^p - d_3}{y_w^p - d_2} f. \tag{11}$$

From (10) and (11) , x_w^p, y_w^p, z_w^p, $\dot{x}_w^p$, $\dot{y}_w^p$ and $\dot{z}_w^p$ can be found in terms of X_1^p, Y_1^p, X_2^p, Y_2^p and their derivatives. Therefore, from the measurements of (X_1^p, Y_1^p) , (X_2^p, Y_2^p) and optical flow $(\dot{X}_1^p, \dot{Y}_1^p)$, $(\dot{X}_2^p, \dot{Y}_2^p)$, the 3D position (x_w^p, y_w^p, z_w^p) and velocity $(\dot{x}_w^p, \dot{y}_w^p, \dot{z}_w^p)$ can be solved. By defining

$$\eta_1(t) = \dot{x}_w^p(t), \qquad \eta_2(t) = \dot{y}_w^p(t), \qquad \eta_3(t) = \dot{z}_w^p(t)$$

$$\phi_1(t) = \begin{bmatrix} z_w^p(t) \\ -y_w^p(t) \\ 1 \end{bmatrix}, \quad \phi_2(t) = \begin{bmatrix} -z_w^p(t) \\ x_w^p(t) \\ 1 \end{bmatrix}, \quad \phi_3(t) = \begin{bmatrix} y_w^p(t) \\ -x_w^p(t) \\ 1 \end{bmatrix},$$

$$\theta_1 = \begin{bmatrix} \omega_2 \\ \omega_3 \\ a' \end{bmatrix}, \quad \theta_2 = \begin{bmatrix} \omega_1 \\ \omega_3 \\ b' \end{bmatrix}, \quad \theta_3 = \begin{bmatrix} \omega_1 \\ \omega_2 \\ c' \end{bmatrix}.$$

(9) is of the form

$$\eta_i(t) = \phi_i(t)^T\theta_i. \quad (i = 1, 2, 3) \tag{12}$$

Hence the standard recursive least-squares estimation [12] assumes the following form:

$$\begin{aligned}
\hat\theta_i(t) &= \hat\theta_i(t-\Delta t) + K_i(t)(\eta_i(t) - \phi_i(t)^T\hat\theta_i(t-\Delta t)), && (13)\\
K_i(t) &= P_i(t-\Delta t)\phi_i(t)(1 + \phi_i(t)^T P_i(t-\Delta t)\phi_i(t))^{-1}, && (14)\\
P_i(t) &= (I - K_i(t)\phi_i(t)^T)P_i(t-\Delta t) && (15)
\end{aligned}$$

where $\hat\theta_i(t)$ and $\hat\theta_i(t-\Delta t)$ are respectively the estimates of θ_i from images measured at time instant t and $t - \Delta t$.

3.2 The Design of Tracking Law

For robot plant (5), (6) to track the estimated exosystem (9), the tracking error is defined as

$$e_r = \begin{bmatrix} x_w^e \\ y_w^e \\ z_w^e \end{bmatrix} - \begin{bmatrix} x_w^p \\ y_w^p \\ z_w^p \end{bmatrix} = C\xi - \begin{bmatrix} x_w^p \\ y_w^p \\ z_w^p \end{bmatrix}. \tag{16}$$

To solve the SFRP for (5), (6), (9) and (16), we define

$$\begin{bmatrix} \tilde x \\ \tilde y \\ \tilde z \end{bmatrix} = \begin{bmatrix} x_w^p \\ y_w^p \\ z_w^p \end{bmatrix} - \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \tilde\xi = \xi - [a\ 0\ b\ 0\ c\ 0]^T$$

where $[a\ b\ c]^T$ is any vector satisfying (9), then

$$\dot{\tilde\xi} = A\tilde\xi + Bv, \quad \begin{bmatrix} \dot{\tilde x} \\ \dot{\tilde y} \\ \dot{\tilde z} \end{bmatrix} = \Omega \begin{bmatrix} \tilde x \\ \tilde y \\ \tilde z \end{bmatrix}, \quad e_r = C\tilde\xi - \begin{bmatrix} \tilde x \\ \tilde y \\ \tilde z \end{bmatrix}.$$

Since the matrix equations $\Pi\Omega = A\Pi + B\Gamma$, $C\Pi - I = 0$ are solvable for Π and Γ, the tracking law

$$v = \Gamma \begin{bmatrix} \tilde x \\ \tilde y \\ \tilde z \end{bmatrix} + K(\tilde\xi - \Pi \begin{bmatrix} \tilde x \\ \tilde y \\ \tilde z \end{bmatrix}) \tag{17}$$

where K satisfies $\sigma(A + BK) \subset \mathbf{C}^-$, will render the closed loop stable as well as $e_r \to 0$ when $t \to \infty$.

(17) is called the position-based tracking law.

4. IMAGE-BASED TRACKING

In the visual feedback tracking literature for robot with eye-on-hand configuration, it is shown that image-based control scheme is more stable than the conventional position-based one which requires the estimation of 3D position of the object. Although the position-based scheme in our setup gives a good simple tracking law, it is still desirable to understand how the image-based tracking is performed and what advantage it may have.

In the RCF with its origin at o_r:(a, b, c), the rotation of the centroid P:(x_r^p, y_r^p, z_r^p) lies in a plane with shape parameters ω_1, ω_2, ω_3 and s, i.e., the coefficients of the plane

$$\omega_1 x_r^p + \omega_2 y_r^p + \omega_3 z_r^p + s = 0. \tag{18}$$

In CCF1 and CCF2, by (1) and (8),

$$\begin{bmatrix} x^p_{c1} \\ y^p_{c1} \\ z^p_{c1} \end{bmatrix} = \begin{bmatrix} -x^p_r - a + d_1 + d \\ z^p_r + c - d_3 \\ y^p_r + b - d_2 \end{bmatrix}, \quad \begin{bmatrix} x^p_{c2} \\ y^p_{c2} \\ z^p_{c2} \end{bmatrix} = \begin{bmatrix} -x^p_r - a + d_1 \\ z^p_r + c - d_3 \\ y^p_r + b - d_2 \end{bmatrix}. \tag{19}$$

So (18) becomes

$$-\omega_1 x^p_{ci} + \omega_3 y^p_{ci} + \omega_2 z^p_{ci} + s_i = 0 \qquad (i = 1, 2) \tag{20}$$

where $s_i = \omega_1(-a + d_1 + d^{(i)}) + \omega_2(-b + d_2) + \omega_3(-c + d_3) + s, \;\; d^{(1)} = d, \;\; d^{(2)} = 0$.

The 3D motion (8) (or (9)) induces two projected motions respectively in the image planes in the following way. (To simplify notation, the image of the centroid P is denoted by (X_i, Y_i) rather than (X^p_i, Y^p_i).)

First notice that by (7) and (20), we get

$$\frac{1}{z^p_{ci}} = \frac{\omega_1 x^p_{ci} - \omega_3 y^p_{ci} - \omega_2 z^p_{ci}}{s_i z^p_{ci}} = \frac{\omega_1 X_i - \omega_3 Y_i - \omega_2}{f s_i}. \tag{21}$$

Then by diffentiating both sides of the equations in (7) with respect to t and using (19) and (21), we obtain the following as the dynamics of the 2D motion in each image plane:

$$\begin{bmatrix} \dot{X}_i \\ \dot{Y}_i \end{bmatrix} = \begin{bmatrix} S_{i1}(X_i, Y_i) \\ S_{i2}(X_i, Y_i) \end{bmatrix}, \quad (i = 1, 2) \tag{22}$$

where

$$S_{i1} = a^{(i)}_{11} + a^{(i)}_{12} X_i + a^{(i)}_{13} Y_i + X_i(a^{(i)}_{14} X_i + a^{(i)}_{15} Y_i),$$
$$S_{i2} = a^{(i)}_{21} + a^{(i)}_{22} X_i + a^{(i)}_{23} Y_i + Y_i(a^{(i)}_{14} X_i + a^{(i)}_{15} Y_i),$$

The above $a^{(i)'}_{jk}s$ are called the projected 2D motion parameters.

4.1 Estimation of Projected Motion Parameters

Since $(\dot{X}_i, \dot{Y}_i)$ and (X_i, Y_i) can be measured from the image plane, the recursive least squares method can be used to estimate the projected 2D motion parameters from (22).

4.2 The Design of Tracking Law in Image Space

It is not hard to see that the tracking of the object by the robot in 3D space can be achieved from the requirement that the end-effector image track the projected motion of the exosystem (22) for each of the two cameras. However, due to the nonlinearities existing in the exosystem and the error function, a set of PDE must be solved in order to find the tracking law. In general, the PDE can only be numerically solved, which increases the computational load and usually makes the real-time implementation of the tracking law impossible. It is thus highly desirable that a closed form solution exist to the resulting PDE. In the following, we show that the objective can be achieved with a suitable definition of the tracking error function.

The images of the end-effector e are (X^e_1, Y^e_1) and (X^e_2, Y^e_2) with

$$X^e_1 = \frac{-x^e_w + d_1 + d}{y^e_w - d_2} f, \quad X^e_2 = \frac{-x^e_w + d_1}{y^e_w - d_2} f, \quad Y^e_1 = Y^e_2 = \frac{z^e_w - d_3}{y^e_w - d_2} f.$$

Define the error function

$$e_r = h(\xi) + q(X_1, Y_1, X_2, Y_2) \tag{23}$$

where

$$h(\xi)=\begin{bmatrix}(-\xi_1+d_1+d)f/(\xi_3-d_2)\\(-\xi_1+d_1)f/(\xi_3-d_2)\\(\xi_5-d_3)f/(\xi_3-d_2)\end{bmatrix},\quad q(X_1,Y_1,X_2,Y_2)=\begin{bmatrix}-X_1\\-X_2\\-(Y_1+Y_2)/2\end{bmatrix},$$

then the SFRP with plant (5), exosystem (22) and error function (23) will result in a PDE having a closed form solution.

To solve the regulator problem, let $(\bar{X}_1,\bar{Y}_1,\bar{X}_2,\bar{Y}_2)$ be the equilibrium point of (22) and let $[\gamma_1\ \gamma_2\ \gamma_3]^T$ be the unique solution of

$$\begin{bmatrix}f&\bar{X}_1&0\\f&\bar{X}_2&0\\0&-(\bar{Y}_1+\bar{Y}_2)/2&f\end{bmatrix}\begin{bmatrix}\gamma_1\\\gamma_2\\\gamma_3\end{bmatrix}=\begin{bmatrix}(d_1+d)f+d_2\bar{X}_1\\d_1f+d_2\bar{X}_2\\d_3f-d_2(\bar{Y}_1+\bar{Y}_2)/2\end{bmatrix}.$$

By changes of coordinates

$$\begin{bmatrix}\tilde{X}_1\\\tilde{Y}_1\\\tilde{X}_2\\\tilde{Y}_2\end{bmatrix}=\begin{bmatrix}X_1\\Y_1\\X_2\\Y_2\end{bmatrix}-\begin{bmatrix}\bar{X}_1\\\bar{Y}_1\\\bar{X}_2\\\bar{Y}_2\end{bmatrix},\quad \tilde{\xi}\triangleq\begin{bmatrix}\tilde{\xi}_1\\\tilde{\xi}_2\\\tilde{\xi}_3\\\tilde{\xi}_4\\\tilde{\xi}_5\\\tilde{\xi}_6\end{bmatrix}=\xi-\begin{bmatrix}\gamma_1\\0\\\gamma_2\\0\\\gamma_3\\0\end{bmatrix},\tag{24}$$

(22), (5) and (23) become respectively

$$\begin{bmatrix}\dot{\tilde{X}}_1\\\dot{\tilde{Y}}_1\\\dot{\tilde{X}}_2\\\dot{\tilde{Y}}_2\end{bmatrix}=\begin{bmatrix}\tilde{S}_{11}(\tilde{X}_1,\tilde{Y}_1)\\\tilde{S}_{12}(\tilde{X}_1,\tilde{Y}_1)\\\tilde{S}_{21}(\tilde{X}_2,\tilde{Y}_2)\\\tilde{S}_{22}(\tilde{X}_2,\tilde{Y}_2)\end{bmatrix}.\tag{25}$$

$$\dot{\tilde{\xi}}=A\tilde{\xi}+Bv.\tag{26}$$

$$e_r=\tilde{h}(\tilde{\xi})+\tilde{q}(\tilde{X}_1,\tilde{Y}_1,\tilde{X}_2,\tilde{Y}_2)\tag{27}$$

where

$$\tilde{h}(\tilde{\xi})=\begin{bmatrix}(-f\tilde{\xi}_1-\bar{X}_1\tilde{\xi}_3)/(\tilde{\xi}_3+\gamma_2-d_2)\\(-f\tilde{\xi}_1-\bar{X}_2\tilde{\xi}_3)/(\tilde{\xi}_3+\gamma_2-d_2)\\(f\tilde{\xi}_5-0.5(\bar{Y}_1+\bar{Y}_2)\tilde{\xi}_3)/(\tilde{\xi}_3+\gamma_2-d_2)\end{bmatrix},\ \tilde{q}(\tilde{X}_1,\tilde{Y}_1,\tilde{X}_2,\tilde{Y}_2)=\begin{bmatrix}-\tilde{X}_1\\-\tilde{X}_2\\-(\tilde{Y}_1+\tilde{Y}_2)/2\end{bmatrix}.$$

Clearly, 0 is the equilibrium of system (25), (26), (27). That is, $\tilde{S}(0)=0$, $\tilde{h}(0)=0$, $\tilde{q}(0)=0$. Applying the regulator theory, and observing that $\tilde{h}(\pi(\tilde{X}_1,\tilde{Y}_1,\tilde{X}_2,\tilde{Y}_2))+\tilde{q}(\tilde{X}_1,\tilde{Y}_1,\tilde{X}_2,\tilde{Y}_2)=0$ is equivalent to

$$\begin{aligned}f\pi_1+(\tilde{X}_1+\bar{X}_1)\pi_3+(\gamma_2-d_2)\tilde{X}_1&=0,\\f\pi_1+(\tilde{X}_2+\bar{X}_2)\pi_3+(\gamma_2-d_2)\tilde{X}_2&=0,\\(\tilde{Y}_1+\bar{Y}_1+\tilde{Y}_2+\bar{Y}_2)\pi_3-2f\pi_5+(\gamma_2-d_2)(\tilde{Y}_1+\tilde{Y}_2)&=0,\end{aligned}$$

we can obtain π_1, π_3, π_5. The analytical mappings $\pi = [\pi_1\ \pi_2\ \pi_3\ \pi_4\ \pi_5\ \pi_6]^T$ and $c = [c_1\ c_2\ c_3]^T$ can thus be solved successively from the PDE

$$\begin{bmatrix} \pi_2(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \\ c_1(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \\ \pi_4(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \\ c_2(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \\ \pi_6(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \\ c_3(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) \end{bmatrix} = \begin{bmatrix} \partial\pi_1/\partial\tilde{X}_1 & \partial\pi_1/\partial\tilde{Y}_1 & \partial\pi_1/\partial\tilde{X}_2 & \partial\pi_1/\partial\tilde{Y}_2 \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \partial\pi_6/\partial\tilde{X}_1 & \partial\pi_6/\partial\tilde{Y}_1 & \partial\pi_6/\partial\tilde{X}_2 & \partial\pi_6/\partial\tilde{Y}_2 \end{bmatrix} \begin{bmatrix} \tilde{S}_{11}(\tilde{X}_1, \tilde{Y}_1) \\ \tilde{S}_{12}(\tilde{X}_1, \tilde{Y}_1) \\ \tilde{S}_{21}(\tilde{X}_2, \tilde{Y}_2) \\ \tilde{S}_{22}(\tilde{X}_2, \tilde{Y}_2) \end{bmatrix}.$$

The tracking law is thus given by

$$v = c(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2) + K(\tilde{\xi} - \pi(\tilde{X}_1, \tilde{Y}_1, \tilde{X}_2, \tilde{Y}_2)). \tag{28}$$

(28) is called the image-based tracking law.

5. SIMULATION STUDY

Our simulation is based on the following specifications: camera focal length $f = 50$ mm, the distance between cameras $d = 0.8$ m. $[d_1\ d_2\ d_3]^T = [0.1\ 4.0\ 0.5]^T$, $[a\ b\ c]^T = [-0.1\ \ -1.5\ \ -1.4]^T$, $(\omega_1,\ \omega_2,\ \omega_3) = (0.1,\ 0.2,\ 0.3)$. The camera's image plane is of the size 0.01(m)×0.01(m), and is digitized to 512 × 512 pixels. The object's centroid is moving circularly in the robot workspace which is approximately 4 meters away from both cameras, with the radius of rotation 0.25 meter. The speed of rotation is about 4 cycles per minute which means the object moves at velocity 10 cm/s.

In position based (or image-based) scheme, motion (or projected motion) parameters are recursively estimated on-line. This keeps updating the control gain in the feedback tracking law (17) (or (28)). The total tracking error at time t is defined to be the distance between the end-effector and the centriod of the moving object at that time. One pixel perturbation is randomly added to the image location of the centroid to simulate the measurement error. Some fixed quantities are added to d, d_1, d_2, d_3 and the measurement of image location to simulate the camera-camera and camera-robot calibration errors.

When there is no calibration error, position-based scheme (Figure 2, 3) and image-based scheme (Figure 4, 5) have comparable tracking performance. Note that in Figure 4 and 8, only the projected motion parameters in image plane 1 are plotted. We can see from Figure 5 and 3 that the tracking error in image-based scheme is slightly bigger than that in position-based one. This is because that the projected motion parameters (see Figure 4) are more difficult to estimate accurately than the 3D motion parameters in our simulation case. (With noisy measurement, estimation of 8 projected motion parameters from a 2D measurement vector is less accurate than estimation of 6 3D motion parameters from a 3D measurement vector). We may expect to improve the estimation of the projected motion parameters by using some observer-based identifier.

With calibration error, we find that the image-based scheme (Figure 8, 9) is much more robust than the position-based one (Figure 6, 7). In fact, after we add some calibration error into image plane 1, the estimates of motion parameters change slightly (Compare Figure 6 with 2, 8 with 4 respectively). Now the image-based scheme has a smaller tracking error than the position-based one (Figure 7 and 9).

6. CONCLUDING REMARKS

In the case of visually-guided robot tracking with one camera, since complete knowledge of the 3D scene can not be obtained, an approximate model is usually used in order to derive the tracking law, and consequently the image-based scheme achieves better tracking performance than the position-based one. When two camera are used, however, the 3D scene information is indeed equivalent to the 2D image information. The resulting position-based and image-based control laws behave similarly for well calibrated robot hand-eye system. However, when perfect calibration is not possible, the image-based scheme gives more robust tracking performance.

References

[1] G. J. Agin, *Real-time Robot Control with a Mobile Camera,* Robotics Today, Society of Manufacturing Engineers, pp. 35-39, Fall 1979.

[2] G. J. Vanderburg, J. S. Albus, and E. Barkmeyer, *A Vision System for Real Time Robot Control,* Robotics Today, Society of Manufacturing Engineers, pp. 20-22, Winter 1979-80.

[3] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, *Dynamic Sensor-Based Control of Robots with Visual Feedback,* IEEE J. of Robotics and Automation, Vol. RA-3, No. 5, Oct. 1987

[4] K. Hashimoto, T. Kimoto, T. Ebine and H. Kimura, *Manipulator Control with Image-Based Visual Servo,* IEEE Int. Conf. Robotics and Automation, p2267-2272, Sacramento, CA, 1991

[5] N. Papanikolopoulos, P. K. Khosla, and T. Kanade, *Vision and Control Techniques for Robotic Visual Tracking,* IEEE Int. Conf. Robotics and Automation, Sacramento, CA, 1991

[6] B. K. P. Horn, *Robot Vision,* Cambridge, Massachusetts, MIT Press, 1986.

[7] P. K. Allen, *Robotic Object Recognition Using Vision and Touch,* Kluwer Academic Publishers, 1987.

[8] P. K. Allen, A. Timcenko, B. Yoshimi, P. Michelman, *Trajectory Filtering and Prediction for Automated Tracking and Grasping of a Moving Object,* Proc. 1992 Int. Conf. Rob. Auto., Nice, France, 1992

[9] B. A. Francis, *The Linear Multivariable Regulator Problem,* SIAM J. Control and Optimization, Vol. 15, pp486-505, 1977.

[10] T. J. Tarn, A. K. Bejczy, X. Yun, *Nonlinear Feedback for PUMA 560 Robot Arm,* SSM-RL-85-03, Dept. SSM, Washington Univ., St. Louis, 1985.

[11] A. Isidori, *Nonlinear Control Systems: An Introduction,* 2nd edition, Heidelberg, Springer-Verlag, 1989.

[12] K. J. Astrom and B. Wittenmark, *Adaptive Control,* Addison-Wesley Publishing Company, 1989.

[13] R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control,* The MIT Press, Cambridge, Massachusetts and London, England, 1981.

[14] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control,* John Wiley and Sons, 1989.

[15] Y. Shirai, *Three-Dimensional Computer Vision,* Springer-Verlag, 1987.

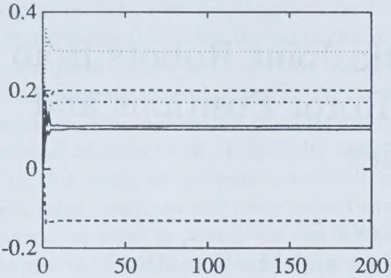


Figure 2: Motion Parameters

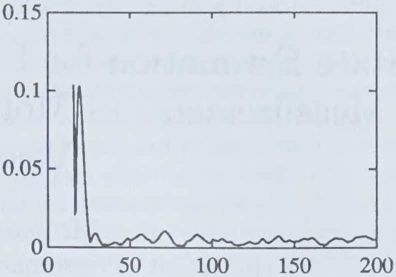


Figure 3: Tracking Error (meter)

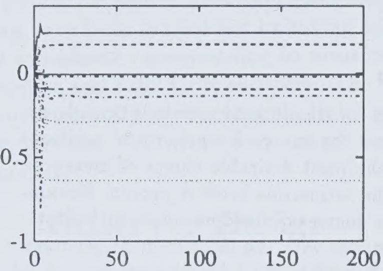


Figure 4: Projected Motion Parameters

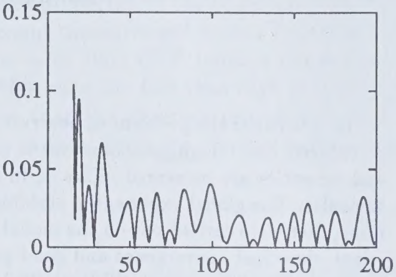


Figure 5: Tracking Error (meter)

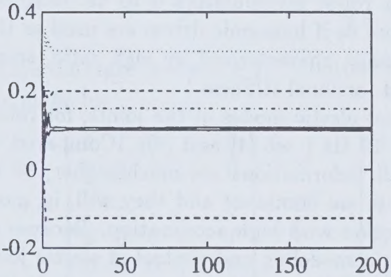


Figure 6: Motion Parameters

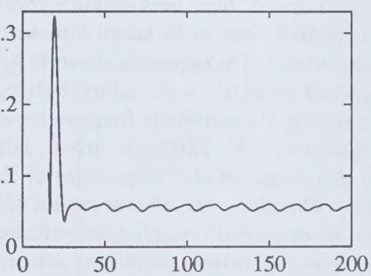


Figure 7: Tracking Error (meter)

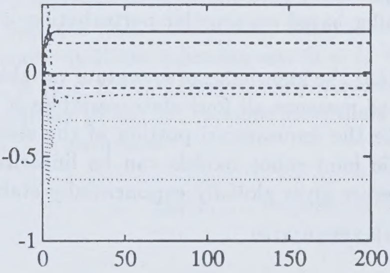


Figure 8: Projected Motion Parameters

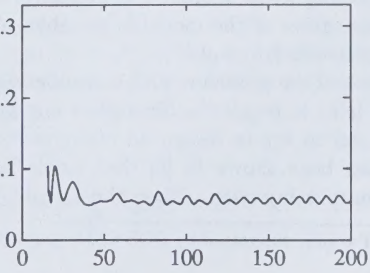


Figure 9: Tracking Error (meter)

State Estimation for Elastic Joint Robots from Measurements of Motor Rotor Positions and Velocities

Mrdjan Jankovic *
Department of Systems Science and Mathematics
Washington University, Campus Box 1040
St. Louis, Missouri 63130

Abstract

In this paper the problem of observer design for elastic joint robots is considered. A reduced order high-gain observer is proposed for the case when rotor positions and velocities are measured. This is, in fact, the most desirable choice of measured variables. The global exponential stability of the estimation error is proven. Simulations, that were carried out on the model of two degrees of freedom robot with elastic joints, show fast convergence and good performance with the measurement accuracy corresponding to commercially available optical encoders.

1 Introduction

If a high speed, high performance controller for robot manipulators is to be designed, joint elasticity has to be taken into account; more so if harmonic drives are used as the transmissions. (A harmonic drive is a transmission characterized by high ratio, small size, small backlash, and, unfortunately, reduced torsional stiffness.)

Typically, characteristic frequencies of torsional elastic modes of the joints, for robot manipulators with harmonic drives, fall bellow 10 Hz (see [4] and [9]). Compared to this, the characteristic frequencies of link elastic deformations are much higher, 30 to 60 Hz. Therefore, the elastic modes of the joints are dominant and they will, in most cases, be excited if the robot is performing a motion with high acceleration. Because of this, there has been a significant research effort in modeling and control of elastic joint robots. In [3] a controller based on dynamic feedback linearization was proposed. In [10] a simplified model of elastic joint robot had been considered and it was shown that by neglecting terms of order 1/m (where m is a transmission ratio) exact static state feedback linearization of the model is possible. A controller based on singular perturbation was proposed in [10] and [11].

One of the problems with controller design is that the knowledge of four state variables per joint is required. Since it is not practical to measure all four state variables it is natural to try to design an observer to estimate the unmeasured portion of the state. It has been shown in [6] that single link elastic joint robot models can be linearized by output injection. Then the Luenberger observer gives globally exponentially stable

---

*Partially supported by DOE under grant No. DE-FG02-90ER14140.

estimates of the states. An approximate observer has been considered in [7] for more general models with displacement between links and corresponding velocities being the measured variables. For the same choice of measured variables a globally asymptotically stable observer has been obtained in [12], a Kalman filter based observer in [5], and a robust discrete time observer in [2]. For link and rotor displacements being the measured variables, the application of a high gain observer was considered in [8] and [13]. The problem of estimation of elasticity constants is also considered in these two references.

In this work we propose a reduced order high-gain observer for elastic joint robots with rotor positions and rotor velocities assumed the measured variables. Stability of the estimation error is proven for the simplified model of elastic joint robots given in [10]. Despite the fact that the above choice of measured variables is the most desirable, to the best of the author's knowledge, observer design in this case was not considered in the literature before.

Simulation results show very good convergence properties of the observer. The simulations have been carried out by taking into account measurement noise. Two noise levels are considered corresponding to measurements with 1000 CPR (counts per revolution) encoders and 8192 CPR encoders. The plots illustrate the fact that high gain observers require accurate measurements, but also show that the observer proposed in this paper performs well with measurement accuracy within limits for commercially available encoders.

2 Observer Design

In this section we present a result that will be used latter to design an observer for elastic joint robots. This result is a modification of the high gain observer proposed in [1].

Consider the nonlinear system in the following form:

$$\begin{aligned} \dot{x} &= Ax + g(x,u) \\ y &= Cx \end{aligned} \tag{2.1}$$

where the state vector $x \in R^{km}$, output vector $y \in R^m$ and the matrices A, C and $g(x,u)$ are given by

$$A = \begin{bmatrix} 0 & I & 0 & \dots & 0 \\ 0 & 0 & I & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & I \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}, \quad C^T = \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad g(x,u) = \begin{bmatrix} \psi_1(x_1,u) \\ \psi_2(x_1,x_2,u) \\ \vdots \\ \psi_{k-1}(x_1,\dots,x_{k-1},u) \\ \psi_k(x,u) \end{bmatrix} \tag{2.2}$$

with the vector x partitioned into $x = \begin{bmatrix} x_1 & x_2 & \dots & x_k \end{bmatrix}^T$, and with all identity and zero matrices being of dimension $m \times m$.

Suppose that there exist a constant $M > 0$ such that $\|x(t)\| < M$ for every $t > 0$. Denote by Ω the following set: $\Omega = \{x \in R^{km} : \|x(t)\| < M\}$. For $\gamma > 1$ define the set $\Omega_\gamma = \{x \in R^{km} : \|x(t)\| < \gamma M\}$. Assume that the functions $\psi_i(x,u)$ satisfy the Lipschitz condition in Ω_γ, i.e. assume that there exist positive constants α_{ij}, $i = 1,\dots,k$, $j = 1,\dots,i$, such that

$$\|\psi_i(x_1,\dots,x_i,u) - \psi_i(z_1,\dots,z_i,u)\| < \sum_{j=1}^{i} \alpha_{ij}\|x_j - z_j\| \tag{2.3}$$

for all $x, z \in \Omega_\gamma$, uniformly in u. For a positive constant G define a matrix Γ via

$$\Gamma = \text{block diagonal}\{GI, G^2 I, \ldots, G^k I\}$$

and consider the following discontinuous observer for the system (2.1).

$$\begin{aligned} \dot{\hat{x}} &= A\hat{x} + g(y, \hat{x}_2, \ldots, \hat{x}_k, u) + \Gamma K(y - C\hat{x}) \\ \hat{x}(t_i^+) &= M\frac{\hat{x}(t_i^-)}{\|\hat{x}(t_i^-)\|} \end{aligned} \tag{2.4}$$

with the sequence of times t_i defined via

$$t_i = \min\{t : \; t > t_{i-1} \text{ and } \|\hat{x}(t)\| \geq \gamma M\}, \quad t_0 = 0$$

We assume that the matrix K is chosen such that the eigenvalues of $A - KC$ are in the open left half of the complex plane. Now for the observer (2.4) the following theorem, which is a modification of Theorem 1 in [1], gives the exponential stability of the estimation error e, defined by $e = x - \hat{x}$.

Theorem 2.1: Suppose that the state of the nonlinear system (2.1) remains in Ω and that the functions $\psi_i, i = 1, \ldots, k$ satisfy the Lipschitz condition (2.3) in Ω_γ. Then there exists a positive constant G_0 such that the estimation error converges to zero exponentially if the constant G in Γ satisfies $G > G_0$.

Proof: Consider first the error dynamics between discontinuities given by

$$\dot{e} = (A - \Gamma KC)e + g(x, u) - g(x^*, u) \tag{2.5}$$

where x^* denotes the vector $\hat{x}$ with $\hat{x}_1$ substituted by y.

Define the matrix P as a positive definite solution of the following Liapunov matrix equation

$$(A - KC)^T P + P(A - KC) = -I$$

Since, by assumption, the matrix $A - KC$ has all its eigenvalues in the open left half of the complex plane, such a solution exists.

Consider the linear coordinate transformation $\eta = \Gamma^{-1}e$. For the system in the new η coordinates define a Liapunov function via $V(\eta) = \eta^T P \eta$. Following the proof of the Theorem 1 of [1] we obtain the upper bound on the time derivative of the Liapunov function $V(\eta)$:

$$\dot{V} \leq -G\|\eta\|^2 + 2\|\eta\|\|P\| \sum_{i=2}^{k}\sum_{j=2}^{i} \alpha_{ij} G^{-k+i} \|\eta_i\| \tag{2.6}$$

It is obvious from the above inequality that there exists a positive constant G_0 such that $\dot{V} < 0$ if G is chosen such that $G > G_0$. Therefore, it follows by the Liapunov Direct Method that $\|\eta(t)\|$ decreases exponentially between discontinuities and so deos $\|e(t)\|$.

Since the states of the system (2.1) are assumed bounded and the states of the observer (2.4) are kept bounded it follows that the right hand side of the differential equation in (2.4) is bounded. Therefore the length of time intervals between discontinuities $|t_i - t_{i-1}|$ is greater then a positive constant δ for all i. On every such a time interval the estimation error $e(t)$ converges to zero exponentially and, at a discontinuity, the estimation error satisfies $\|e(t_i^+)\| < \|e(t_i^-)\|$. Therefore we conclude that the estimation error is exponentially stable.

Q.E.D.

3 A high gain observer for elastic joint robots

In this section a reduced order high-gain observer is proposed for a model of elastic joint robot adopted form [10]. The model represents an n degrees of freedom manipulator with elasticity in the joints modeled as torsional springs. It has been obtained under the assumption that the motion of the motor rotors is pure rotation with respect to an inertial frame; in other words the terms of order $1/m$ are neglected. For motivation via an example see [10].

The vector of generalized coordinates q is partitioned into two groups: $q = [q_1\ q_2]^T$; where q_1 represents total displacements between links (actuator (rotor) plus elastic displacements) and q_2 represents rotor displacements. If the friction forces are neglected the model takes the following form:

$$\begin{aligned} &B(q_1)\ddot{q}_1 + C(q_1, \dot{q}_1)\dot{q}_1 + K_e(q_1 - q_2) + e(q_1) = 0 \\ &J\ddot{q}_2 - K_e(q_1 - q_2) = u \end{aligned} \tag{3.1}$$

where $B(q_1)$ is symmetric, uniformly positive definite and bounded inertia matrix; J is the diagonal matrix of rotor inertia reflected to the link side of the transmission; K_e is the diagonal matrix of elasticity constants and u is the vector of input torques. Terms $C(q_1, \dot{q}_1)\dot{q}_1$ and $e(q_1)$ in (3.1) represent torques due to centrifugal and Coriolis forces and gravitational forces respectively.

Defining the state variables x_i as $x_1 = q_1$, $x_2 = \dot{q}_1$, $x_3 = q_2$ and $x_4 = \dot{q}_2$ we obtain the following state space model:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -B^{-1}(x_1)K_e(x_1 - x_3) - B^{-1}(x_1)C_1(x_1, x_2)x_2 - B^{-1}(x_1)e(x_1) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= J^{-1}K_e(x_1 - x_3) + J^{-1}u \end{aligned} \tag{3.2}$$

Suppose that a robot manipulator is designed with speed and position sensors (typically rotary encoders) mounted on the motor shafts. If the motor rotor speeds and positions are measured variables we shall consider a reduced order observer based on the high gain observer considered in Section 2. Recall that the matrix $J^{-1}K_e$ is constant and diagonal and consider the following change of coordinates:

$$\begin{aligned} z_1 &= K_e^{-1}Jx_4 \\ z_2 &= x_1 \\ z_3 &= x_2 \end{aligned} \tag{3.3}$$

The reduced order system obtained via the above change of coordinates satisfies the following differential equations:

$$\begin{aligned} \dot{z}_1 &= z_2 - y_1 + K_e^{-1}u \\ \dot{z}_2 &= z_3 \\ \dot{z}_3 &= \psi(z_2, z_3, y_1) \\ y_2 &= J^{-1}K_e z_1 \end{aligned} \tag{3.4}$$

where $y_1 = x_3$ and $\psi(z_2, z_3, y_1) = B^{-1}K_e(y_1 - z_2) - B^{-1}C_1z_3 - B^{-1}e_1$.

Denote by A_3 and C_3 a pair of matrices in block observer canonical form, i.e. A_3 and C_3 are given by (2.2) with $k = 3$. Then the above system (3.4) can be written in the following form:

$$\begin{aligned} \dot z &= A_3 z + g(z, y_1, u) \\ y_2 &= J^{-1} K_e C_3 z \end{aligned} \tag{3.5}$$

with $g(z, y_1, u) = \left[\begin{array}{ccc} (K_e^{-1}u - y_1)^T & 0 & \psi(z_2, z_3, y_1)^T \end{array} \right]^T$.

Note that the $J^{-1}K_e$ factor in the output equation prevents (3.5) to be in the form (2.1). In addition, in order to use Theorem 2.1, we must require boundedness of the states of the system. Here we will assume a slightly weaker condition which is equivalent to assuming that the total energy of the elastic joint robot is bounded.

Assumption 3.1: Suppose there exist two positive constants M and N such that the following two conditions hold for the system (3.2):

1. $\|x_2\| < M$

2. $\|x_1 - x_3\| < N$

Now following the high gain observer design procedure of Section 2, for a positive constant G, we denote by Γ_3 the following matrix:

$$\Gamma_3 = \text{block diagonal}\{GI, G^2 I, G^3 I\}$$

and we choose a matrix K_3 that stabilizes $A_3 - K_3 C_3$. Define the matrix $\bar K_3 = K_3 K_e^{-1} J$ and consider the following discontinuous observer for the system (3.5):

$$\begin{aligned} \dot{\hat z} &= A_3 \hat z + g(z^*, y_1, u) + \Gamma \bar K_3 (y_2 - J^{-1} K_e C_3 \hat z) \\ \hat z_3(t_i^+) &= M \frac{\hat z_3(t_i^-)}{\|\hat z_3(t_i^-)\|} \\ \hat z_2(s_i^+) &= \left(y_1 - N \frac{y_1 - \hat z_2}{\|y_1 - \hat z_2\|} \right)\Big|_{s_i^-} \end{aligned} \tag{3.6}$$

with the sequences of times t_i and s_1 defined via

$$\begin{aligned} t_i &= \min\{t : \ t > t_{i-1} \text{ and } \|\hat z_3(t)\| \geq \gamma M\}, \quad \text{with } \ t_0 = 0 \\ s_i &= \min\{s : \ s > s_{i-1} \text{ and } \|y_1 - \hat z_2(t)\| \geq \gamma N\}, \quad \text{with } \ s_0 = 0 \end{aligned}$$

and with $z^* = \left[\begin{array}{ccc} y_2^T & \hat z_2^T & \hat z_3^T \end{array} \right]^T$.

The estimation error $e = z - \hat z$ satisfies, between discontinuities, the following differential equation:

$$\dot e = (A_3 - \Gamma_3 K_3 C_3) e + g(z, y_1, u) - g(z^*, y_1, u) \tag{3.7}$$

Now the following theorem gives the exponential stability of the estimation error.

Proposition 3.2: Suppose that the Assumption 3.1 is satisfied. Then there exists a positive constant G_0 such that the estimation error dynamics (3.7) is globally exponentially stable provided the constant G in the matrix Γ_3 satisfies $G > G_0$.

Proof: The proof follows immediately from Theorem 2.1.

4 Simulation results

The simulations have been carried out for the elastic joint robot model adopted from [6]. The model represents two degrees of freedom robot with elastic joints and is given by the following differential equations:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\dot{x}_3 &= \frac{k_1(x_5 - x_1) + h_2 \sin 2x_2 \, x_3 x_4}{h_5 + h_2 \cos^2 x_2} \\
\dot{x}_4 &= h_1 k_2(x_6 - x_2) - h_1(h_6 \cos x_2 - 1/2 h_2 \sin x_2 \, x_3^2) \\
\dot{x}_5 &= x_7 \\
\dot{x}_6 &= x_8 \\
\dot{x}_7 &= \frac{k_1(x_1 - x_5) + u_1}{h_3} \\
\dot{x}_8 &= \frac{k_2(x_2 - x_6) + u_2}{h_4}
\end{aligned} \tag{4.1}$$

where x_1 and x_2 represent link positions, x_3 and x_4 represent link velocities, x_5 and x_6 represent rotor positions and x_7 and x_8 represent rotor velocities. The parameters h_i's and k_i's represent inertia parameters and elasticity constants respectively and they are assumed to have the following values:

$$\begin{array}{lll}
h_1 = 0.095 & h_2 = 22.058 & h_3 = 23.296 \\
h_4 = 70.656 & h_5 = 0.084 & h_6 = 291.106 \\
k_1 = 29,800 & k_2 = 14,100 &
\end{array} \tag{4.2}$$

Note that the values of the elasticity constants are chosen ten times smaller then in [6]. These smaller values better fit into the typical range for elasticity constants for robots with harmonic drives (see for example Section 4.8 in [9]).

It has already been mentioned that the rotor positions and rotor velocities are the most desirable state variables to be measured in elastic joint robots. The reason for this is that both variables can be measured via an encoder mounted on a rotor shaft. This is, in fact, the way how most robots with revolute joints are designed.

Proposition 3.2 states that if the Assumption 3.1 is satisfied and the constant G is sufficiently large then the estimation error converges to zero. However, the question remains whether the effective observer gain (defined to be the norm of the matrix that multiply measurement noise), which is proportional to G^3, is too high for the observer to be applicable. Therefore, in order to show that the observer (3.6) is applicable, we have to consider simultaneously the values of the effective observer gain and the level of the measurement noise.

The effective observer gain is given by $\Gamma_3 K_3 K_e^{-1} J$. In order to guarantee the stability of the estimation error, G has to be chosen to satisfy (see (2.6))

$$G > 2\|P\|(G^{-1}\alpha_2 + \alpha_3)$$

where α_2 and α_3 are Lipschitz constants for $\psi(z_2, z_3, y_1)$ associated with z_2 and z_3 respectively. Note that due to very large values of elasticity constants, $\alpha_2 >> \alpha_3$, the

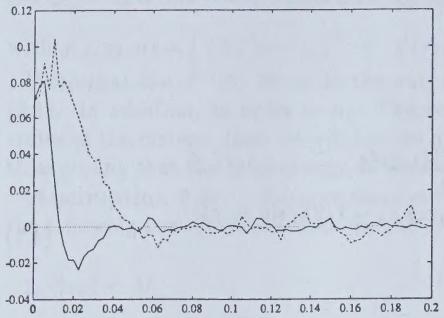


Figure 1: Position estimation errors: 1000 CPR encoders

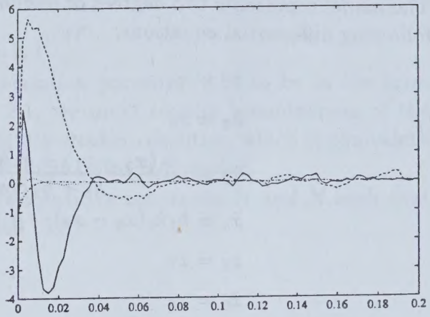


Figure 2: Velocity estimation errors: 1000 CPR encoders

desired value of the constant G is approximately $G = \sqrt{2\|P\|\alpha_2}$. Since the constant α_2 is proportional to the matrix K_e we conclude that the smaller the compliance the larger the gain. It is easy to see that the effective observer gain grows proportionally to $\|K_e\|^{1/2}$.

Assume that the position of the second link remains in the following region: $x_2 \in (-1.3, 1.3) \pm k\pi$ rad. Then, for our example, the two different values of G, corresponding to two joints, have been chosen : 160 and 80. The two different values have been chosen because of the different compliances of the two joints. It can be shown that this does not affect the stability of the observer. The matrix K_3 is chosen to place all the eigenvalues of $A_3 - K_3C_3$ at -1. With this choice of constants the norm of the gain is 3250.

Next we consider the measurement noise. We will assume that the velocity is measured by counting pulses from an encoder in a time interval of known length. The transmission ratio of the drives is assumed to be 150 for both joints. This value is within range of transmission ratios for harmonic drives that goes from 60 to 300 (see [9], page 242). Note that the higher the transmission ratio the lower the output noise level and therefore the better the state estimates. Now we will consider measurement of the velocities with two different encoders; one with 1000 counts per revolution and the other with 8192 counts per revolution. Both encoders fall within the range of accuracy of moderately expensive optical encoders. If the higher accuracy is required, laser encoders have as many as 80,000 CPR, but their price is high. The measurement noise is in both cases modeled as a random process, uniformly distributed between ±0.05rad/s for 1000 CPR encoder and between ±0.006rad/s for 8192 CPR encoder. These values correspond to a sampling frequency of 1600 Hz for velocity measurements (in other words the velocity is proportional to the number of pulses received in 0.625 ms interval).

Maximal velocity of each link is assumed to be 5 rad/s and maximal elastic displacement is assumed 0.08 rad. The constant γ is chosen to be 1.2. The initial condition of the robot contains 0.02 rad elastic displacement for both joints.

The simulation results are plotted in Fig.1-4. Figures 1 and 2 show position and velocity estimation errors for the 1000 CPR encoder measurement. Figures 3 and 4 show position and velocity estimation error for the 8192 CPR encoder. Plots show that the

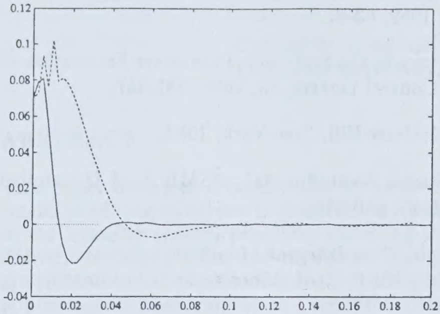


Figure 3: Position estimation errors: 8192 CPR encoders

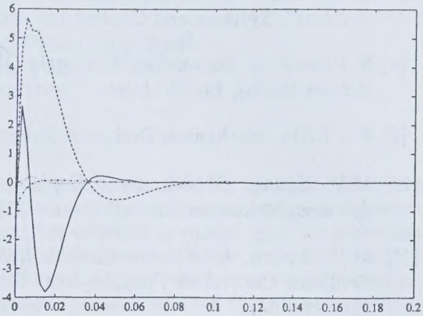


Figure 4: Velocity estimation errors: 8192 CPR encoders

measurement with the 1000 CPR encoder may not be accurate enough (this will depend on how robust a controller is if the observer is used together with a controller). However, when the velocities are measured with 8192 CPR encoders, the observer provides very accurate estimates of the states.

There is a number of advantages of the observer for elastic joint robots proposed in this paper. First, it assumes measurement of the rotor positions and velocities which is the most desirable choice of measured variables. Second, this observer is simple. It does not require nonlinear transformations or propagation of more differential (difference, for discrete time realizations) equations then there are state variables. In fact, because it is a reduced order observer, it requires the propagation of only 3n (n is the number of degrees of freedom) differential (difference) equations. Third, because of the high observer gain, it is robust with respect to changes in inertia parameters. The only disadvantage is the requirement for accurate velocity measurement which may necessitate the use of encoders with large number of counts per revolution.

References

[1] G. Bornard, H. Hammouri, "A High Gain Observer for a Class of Uniformly Observable Systems", Proceedings 30. IEEE CDC, Brighton, England, 1990, 1494-1496.

[2] S.A. Bortoff, J.Y. Hung, M.W. Spong, "A Discrete-Time Observer for Flexible-Joint Manipulators," Proceedings 28th IEEE CDC, Tampa, Florida, 1989, 2078-2082.

[3] A. De Luca, A. Isidori, F. Nicolo,"Control of Robot Arm with Elastic Joints via Nonlinear Dynamic Feedback", Proceedings 24. IEEE CDC, Ft. Lauderdale, FL, 1985, 1671- 1679.

[4] M.C. Good, L.M. Sweet, K.L. Strobel,"Dynamic Models for Control Systems Design of Integrated Robot and Drive Systems", ASME J. of Dynamic Systems, Measurement, and Control, 107, 1985, 53-59.

[5] M. Jankovic, "Exponentially Stable Observer for Elastic Joint Robots," to be presented at the 31st IEEE CDC, Tucson, AZ.

[6] S. Nicosia, P. Tomei, A. Tornambe, "A Nonlinear Observer for Elastic Robots", IEEE Journal Robotics and Automation, 4, 1988, 45-52.

[7] S. Nicosia, P. Tomei, A. Tornambe, "An Approximate Observer for a Class of Nonlinear Systems", Systems and Control Letters, 12,1989, 43-51

[8] S. Nicosia, A. Tornambe, "High-gain Observers in the State and Parameter Estimation of Robots Having Elastic Joints," Systems & Control Letters, 13, 1989, 331-337.

[9] E.I. Rivin, Mechanical Design of Robots, McGraw-Hill, New York, 1988.

[10] M.W. Spong, "Modeling and Control of Elastic Joint Robots", ASME J. of Dynamical Systems, Measurements and Control, 109, 1987, 310-319.

[11] M.W. Spong, K. Khorasani, P.V. Kokotovic, "An Integral Manifold Approach to the Feedback Control of Flexible Joint Robots", IEEE J. of Robotics and Automation, 3, 1987, 291-300.

[12] P. Tomei, "An Observer for Flexible Joint Robots" IEEE Trans. Automatic Control, 35, 1990, 739-743.

[13] A. Tornambe, "Control Law Based on Parameter Observation," ASME J. Dynamic Systems, Measurement, and Control, 114, 1992, 186-195.

INDIRECT LEARNING CONTROL ALGORITHMS FOR TIME-VARYING AND TIME-INVARIANT SYSTEMS[1]

Richard W. Longman[2] and Yeong Soon Ryu[3]
Columbia University, New York, NY 10027

ABSTRACT

This paper develops a framework for indirect learning control based on the type of system model used. An overview is given of the methods of converting indirect adaptive control ideas to the learning control problem, including the possible choices in model type, in repetition domain or mixed time-and-repetition domain implementation, feasible desired trajectory specification, the choice of real-time computations versus apply-when-ready computations, appropriate adjustments when a repetitive disturbance is present, and the significance of time-varying versus time-invariant system models. In the process, a bridge is developed extending indirect adaptive control theory to the more general problems handled by indirect learning control, and techniques are developed to realize the benefits of both adaptive and learning simultaneously.

I. INTRODUCTION

Starting around 1984, there has been considerable research activity in the fields of learning and repetitive control motivated by robot applications. These control algorithms learn from previous experience performing a specific task in order to improve their performance in future executions of this task. References [1-24] give works by the authors and co-workers, which represent a rather comprehensive theory. Reference [1], submitted in November 1984 from Australia, is one of the early independent developments of learning in robotic systems. In [25], Arimoto cites other independent developments in 1984 from other continents: Arimoto, Kawamura, and Miyazaki [26] in Japan, Craig [27] in the US, and Casalino and Bartolini [28] in Italy, in addition to the earlier work of Uchiyama [29] in Japan.

In the usual application of learning control, there is a feedback controller operating that attempts to follow a tracking command. It is the job of the learning control law to learn an additive adjustment to the input command in order to eliminate the errors that remain in the feedback control system output. These errors come from two main sources: 1. First, it is only under very special circumstances that a feedback control system given a command will produce zero error in following the command, even when there are no disturbances and the control system operates perfectly. 2. Secondly, when there are disturbances such as gravity torques on a robot link, these disturbances will often be repetitive, appearing every time the robot performs the maneuver. The learning control laws eliminate such deterministic tracking errors that repeat every time the command is executed. The learning control problem applies to the case when the initial condition is always the same every time the command is executed. When the command is simply a periodic function of time, the term repetitive control is used for systems that learn to improve performance with periods repeat.

The learning methods in [1-24] are based on control theoretic or nonlinear optimization approaches, which allows one to obtain explicit conditions for convergence, or to actually guarantee the convergence of the learning process. The research can be divided into five main categories: 1. Learning based on integral control concepts applied in the repetition domain, together with substantial extensions of this concept. 2. Learning based on indirect adaptive control concepts applied in repetitions. 3. Learning based on direct adaptive control and model reference ideas. 4. Learning based on iterative numerical optimization methods. 5.

---

[1] Research Supported by NASA Grant NAG 1-649.

[2] Professor of Mechanical Engineering.

[3] Graduate Research Assistant, Department of Mechanical Engineering.

Decentralized learning control algorithms based on the above ideas.

The most basic and predominant form of learning control in the literature is based on integral control concepts, and requires that a stability condition be satisfied to insure that the learning process converges to zero tracking error as the repetitions of the task progress. Knowing that this criterion is satisfied requires some knowledge of the system, i.e. the discrete time input and output matrices. References [4] and [5] develop adaptive control based learning control algorithms that produce guaranteed convergence of the learning process in both linear time-invariant and in linear time-varying systems with still weaker requirements about knowledge of the system. Adaptive control theory can be divided into direct and indirect adaptive control, depending on whether adaptive control action requires simultaneous identification of the system. By analogy, references [4] and [5] develop direct learning control and indirect learning control algorithms.

Reference [5] develops one form of indirect learning control. In generating decentralized indirect learning control algorithms, reference [20] develops other possible methods, some of which make use of the identification in the repetition domain as discussed in [12]. It is the purpose of this paper to develop an overview of the methods of using indirect adaptive control ideas on the learning control problem, as related to the mathematical formulation of the feedback system in both the repetition and time domains. In the process, we will develop a bridge connecting indirect adaptive control to indirect learning control, and we will develop ways to simultaneously realize the benefits offered by both adaptive and learning control.

II. SYSTEM MODELS

In this paper we handle a hierarchy of possible mathematical models for the feedback system to which we add a learning control loop. For purposes of illustration, consider a simple example of commanding a single link of a robot arm to execute a desired trajectory in the vertical plane. Let the joint variable be $\theta(t)$ measured from the horizontal, and its commanded time history be $\theta^*(t)$. Let the feedback controller for this joint be a PD controller, so that the system equations and control action are written respectively as

$$\begin{aligned} I\ddot{\theta} + Kg\cos\theta &= v \\ v &= -k_1(\theta - \theta^*) - k_2(\dot{\theta} - \dot{\theta}^*) + u \end{aligned} \tag{1}$$

where g is the gravitational constant. For a general time varying desired trajectory, the system response θ will not equal the commanded response $\theta^*(t)$. This is true of the PD controlled system without the influence of gravity, and gravity further disturbs the system response. It is the job of the learning control u to eliminate both of these repeatable errors, by making use of previous experience at performing the command. The learning control algorithms developed in [5,20] handle system (1) by modelling the system as linearized about the desired trajectory

$$\begin{aligned} I\ddot{\theta} + k_2\dot{\theta} + [k_1 - Kg\sin\theta^*(t)]\theta = k_1\theta^*(t) + k_2\dot{\theta}^*(t) + u \\ -Kg(\sin\theta^*(t))\theta^*(t) - Kg\cos\theta^*(t) \end{aligned} \tag{2}$$

Note that linearizing about a time varying trajectory has introduced time varying coefficients in the resulting linearized equation. Also, the 0th order effect of the gravitational disturbance torque has produced the last terms in the equation, which represents an exogenous forcing function that repeats every time the control task is performed. Making use of the state transition matrix for time varying linear systems, and the convolution integral solutions over each time step, this equation can be converted to digital form without approximation

$$\begin{aligned} x((k+1)T) &= A(k)x(kT) + B(k)u(kT) + w(k) \\ y(kT) &= C(k)x(kT) \end{aligned} \tag{3}$$

where *w(k)* is a digital periodic disturbance. In the sequel, for simplicity we will drop explicit mention of the dependence on time step T in the notation.

The learning control problem is then to find a learning control sequence $u(0), u(1), \ldots, u(p\text{-}1)$ for the p step process, that causes the system (3) to produce the desired output sequence $y^*(1), y^*(2), \ldots, y^*(p)$, without prior knowledge of the system matrices. Knowledge of an upper bound on the order of the system is assumed.

In this paper we first consider indirect learning control for such time varying linear difference equations having a repetitive disturbance. Different possible representations of the system in the repetition domain are considered, including a modern control form and a higher order difference equation form. And methods of combining learning and adaptive control are presented that realize the benefits of each to the extend possible.

The paper proceeds to three more classes of problems. The repetitive disturbance is deleted from the model, as can be appropriate for a large class of systems, and the changes in the way one applies learning control are noted. This is followed by considering the class of problems having constant coefficients, but with a repetitive disturbance, and then the final class of models considered is linear difference equations with constant coefficients and without a repetitive deterministic disturbance. It will be seen that the logical method of applying learning control to this final simplest class of systems is only one step removed from indirect adaptive control, provided the on-line computation ability can keep up with the computations, and hence the bridge between learning and adaptive control is completed.

III. ADAPTIVE CONTROL VERSUS LEARNING CONTROL

In this section we make a preliminary comparison of indirect learning control as presented in [5] and indirect adaptive control for linear systems (certain recent indirect adaptive control methods for nonlinear systems and the associated learning control versions will be discussed in a separate paper). First we make the following points about indirect adaptive control:

1. The strongest convergence results are for linear time-invariant systems. However, the most common situation where there is a need for adaptive control is one in which the system varies with time or is nonlinear.
2. Adaptive control *cannot produce zero tracking error* in a time varying system, because of the time needed to adapt to changed conditions.
3. Adaptive control *may require significant on-line computation ability.*
4. Adaptive control *cannot handle deterministic disturbances* without attempting to model them, often with an ad hock model.

In contrast to these points, indirect learning control uses the fact that the same task is being repeated to accomplish the following:

1. Indirect learning control *can guarantee convergence to zero tracking error* in time varying systems -- the adaptation time lag of item 2 above is eliminated.
2. Learning control *can handle deterministic disturbances* that occur every repetition, and do so very easily.
3. Learning control has *no on-line computation constraint.* Whenever the computations are complete, they can be applied to the next repetition of the command.

The reason why learning control can accomplish these items can be explained easily. Learning control operates in the repetition domain rather than in the time domain, and the same variations of the equation coefficients are observed every repetition. A tracking problem in the time domain with a time varying system, becomes a *regulator* problem in the repetition domain, with a *repetition invariant system.* This allows one to make the strong results of adaptive theory for time invariant systems apply to time-varying systems in the learning control laws. Also, a deterministic disturbance that appears every repetition, can be eliminated from the equations by differencing the equations for successive repetitions, and then using adaptive ideas to solve for the *change* in the control for the next repetition.

On the other hand, in some situations indirect adaptive control does have some advantages over indirect learning control as formulated in [5] as a batch process for each repetition. It can adapt in real time, whereas, the indirect learning control in [5] only learns by comparing results from successive repetitions. Thus, adaptive control may be able to *approach* zero tracking error sooner than the learning control of [5]. One of the main objectives of this paper is to develop modified learning control algorithms that combine the advantages of adaptive control with those of learning control, so that the learning is accomplished at the earliest possible time. How this combination of adaptive and learning control is accomplished is a function of the type of system being controlled, and the way in which it is modelled.

IV. TIME-VARYING SYSTEMS WITH A REPETITIVE DISTURBANCE

The indirect adaptive control techniques considered here, make use of one step ahead control together with recursive on-line identification using a parameter/data equation of the form $y = \phi^{\mathrm{T}}\theta$ [30]. There exist various ways to develop this form for use in the learning control problem.

1. A Deterministic Moving Average (DMA) Model in the Repetition Domain: Reference [5] used the modern control theory formulation of (3) to form a modern control model in the repetition domain, $\underline{y}_r = I\underline{y}_{r-1} + P\delta_r\underline{u}$, whose only unknown is the input influence matrix P. Let the difference operator δ_r operating on any quantity represent the value of that quantity at repetition r minus the value at repetition r-1. Since $w(k)$ repeats each repetition, and since in the learning control problem it is assumed that the initial condition is the same every repetition, using such differences between successive repetitions eliminates these quantities from the equations and we can rewrite (3) for the whole history of a repetition as

$$\delta_r\underline{y} = P\delta_r\underline{u} \tag{4}$$

$$\underline{y} = \left[y^T(1) \quad y^T(2) \quad \cdots \quad y^T(p)\right]^T$$

$$\underline{u} = \left[u^T(0) \quad u^T(1) \quad \cdots \quad u^T(p-1)\right]^T$$

$$P = \begin{bmatrix} C(1)B(0) & 0 & \cdots & 0 \\ C(2)A(1)B(0) & C(2)B(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C(p)\left(\prod_{k=1}^{p-1} A(k)\right)B(0) & C(p)\left(\prod_{k=2}^{p-1} A(k)\right)B(1) & \cdots & C(p)B(p-1) \end{bmatrix} \tag{5}$$

The product notation represents a matrix product going from larger arguments on the left to smaller arguments on the right. Then for the recursive parameter identification of P, one uses the system equation (4) for all repetitions available

$$\left[\delta_1\underline{y} \quad \delta_2\underline{y} \quad \cdots \quad \delta_r\underline{y}\right] = P\left[\delta_1\underline{u} \quad \delta_2\underline{u} \quad \cdots \quad \delta_r\underline{u}\right] \tag{6}$$

and the one step ahead control law is given by (4) which computes the *change* to make in the control actions in the next repetition.

2. New Implementation Option Available in the Learning Control Problem: One can simply apply the above formulas as a batch process for each repetition. However, there is a new option available that has no analogue in the adaptive control problem -- one can learn in a wave. By the structure of (3), one only needs to know $C(1)B(0)$ in order to produce zero tracking error at time step 1, and then once zero tracking error is achieved at that step, the value of $\delta_1 u(0)$ can be zero in future repetitions. This means that one then only needs to identify $C(2)B(1)$ to achieve zero tracking error at time step 2. Hence, by agreeing to produce zero tracking error one time step at a time, one only needs to identify the block diagonal elements of matrix P. This may produce slower convergence than other approaches, but it does have the advantage that it establishes when the requested desired trajectory is infeasible due to saturation limits on the actuators, and tells precisely what time step asks for too much output from the actuators. The approach of learning in a wave discussed here, is important in various indirect decentralized learning control algorithms presented in [20]. In a rather different context, learning in a wave was discovered to be very helpful in improving the transients in the learning process when using integral control based learning control in [17].

3. The Extra Freedom in Computation Timing in the Learning Control Problem: There is a set of possible choices for the timing of the computations needed in computing the learning control signal. 1. There may be enough time between repetitions of the task so that all needed computation can be made at these times. 2. If it is desired to get started on the computations as soon as possible, one notes that as each new piece of data arrives, a recursive least squares computation of the elements of P can be performed to update the estimate. 3. If these computations can be made in real time, then the one step ahead control from (4) can use the latest estimated value of P during a repetition to compute the control actions for that repetition. In the time-varying case discussed here, this does not help, but it does help in the time-invariant cases discussed below. It allows the computation of the desired change in the control $u(k)$ in terms of updated coefficients times the changes in the controls that have already been applied during the current repetition. 4. On the other hand, if the real time computation ability cannot keep up with the computations, there is no difficulty in the learning control problem. One simply applies the resulting learning control actions on the first repetition after they become available (or perhaps on the first time step after they become available).

4. Under-specification to Guarantee Feasibility of the Desired Trajectory: When there are more output variables in $y(k)$ than input variables $u(k\text{-}1)$, then it is not possible to arbitrarily specify the desired trajectory. This is a second type of infeasibility of the specified desired trajectory. Reference [3] chooses to eliminate the appropriate number of variables from each time step in order to solve this problem, and acknowledges that one does not need to choose to eliminate the same variables each time step, so that one can ask for tracking of all desired variables, but not at every time step. An alternative to this is to specify all desired variables at the same time step, but to skip time steps. For example, if one chooses to specify the desired values of all output variables every other time step, one can write

$$\begin{aligned} x(k+2) &= Ax(k+1)+Bu(k+1)+w(k+1) \\ &= A^2x(k)+\begin{bmatrix} AB & B \end{bmatrix}\begin{bmatrix} u(k) \\ u(k+1) \end{bmatrix}+(Aw(k)+w(k+1)) \\ &= \overline{A}x(k)+\overline{B}\overline{u}(k)+\overline{w}(k) \end{aligned} \tag{7}$$

and then use learning control on the barred system. For simplicity, the system has been given as time invariant, but of course the method applies to time varying systems as well.

The new $\overline{B}$ matrix in (7) becomes the controllability matrix once enough time steps are skipped, and for a controllable time-invariant system of order n with m inputs it has reached full rank once n - m time steps have been skipped. Therefore, when there are more outputs than inputs and one knows that the system is controllable, one can guarantee that the desired trajectory is feasible in the sense discussed here, if one has an upper bound $\overline{n}$ on the order of the system, and one skips $\overline{n}$ - m time steps between each specification of the desired value of the output variables. Note that some of the learning schemes developed here allow one to learn in real time during a repetition, and that when using a model as in (7) or its equivalent deterministic autoregressive moving average model, the learning controls are applied every step, but based on data every $\overline{n}$ - m steps, i.e., for those steps that will achieve zero tracking error.

This procedure will usually under-specify the set of equations, so that there are many learning control histories that can produce zero tracking error. One can then choose the minimum norm solution for the $\overline{n}$ - m step ahead controller by using the Moore-Penrose pseudo-inverse solution of equation (4) as modified by the skip step method for the learning control law. The convergence to zero tracking error is then easily proved as before.

5. Convergence to Minimum Least Squares Error in Following a Nonfeasible Desired Trajectory: Alternatively, one can ignore the issue of specifying a feasible desired trajectory, and simply specify whatever you want the system to do every time step. One cannot guarantee zero tracking error in this case unless the trajectory happens to be feasible. In order to handle the situation when it is not, one uses a one-step ahead control law from equation (4) that again makes use of a Moore-Penrose pseudo-inverse. This time it is used because the equations may be overspecified, and one finds the minimum equation error solution, i.e. one obtains guaranteed convergence to the feasible trajectory that is closest to the desired trajectory in a minimum Euclidian norm sense.

6. Indirect Learning Control Based on a Time-Varying DARMA Model: Equation (3) can be converted to a deterministic autoregressive moving average model

$$A(q^{-1},k)y(k) = B(q^{-1},k)u(k) + C(q^{-1},k)w(k) \tag{8}$$

where the A, B, and C are now time-varying difference operators. Difference this equation for the current repetition with the same equation for the last repetition to eliminate the repetitive disturbance terms, and solve for the most recent value of the output y to obtain the form

$$\begin{aligned} \delta_j y(k+1) &= \alpha_0(k)\delta_j y(k) + \ldots + \alpha_{n-1}(k)\delta_j y(k-n) \\ &\quad + \beta_0(k)\delta_j u(k) + \ldots + \beta_l(k)\delta_j u(k-l) \\ \delta_j y(k+1) &= \delta_j \phi^T(k)\theta(k) \\ k &= 0,1,\ldots,p-1 \end{aligned} \tag{9}$$

where the θ contains the difference equation coefficients. The recursive least squares solution of these equations for all time steps in all repetitions up to the current repetition, $j=1,2,\ldots,r$, forms the identification part of the learning control algorithm. The one step ahead learning control is obtained by substituting the current values of the identified coefficients in the first of equation (9) and solving for the change in the learning control for time step k that according to the present model will produce the change in the output at time step $k+1$ needed to have zero tracking error. One can show, following the proof in [30], that this results in zero tracking error under the assumptions there for adaptive control. In the full length journal version of this paper the proof will be given that

$$\lim_{r\to\infty}\left[y_r(k) - y_r^*(k)\right] = 0 \quad \forall \quad k = 1,2,3,\ldots,p \tag{10}$$

when the learning control discussed here is applied to a time varying system, that is periodically restarted from the same initial condition, and having a repetitive unknown disturbance.

Most all of the discussion in the above subsections IV.2 through IV.5 also applies to this DARMA form of learning control: IV.2. The section discussing a new implementation option can be applied, making the learning progress in a wave that learns to produce zero tracking error at one time step before it progresses to the next time step. In this case the first of equations (9) has all zero elements except for the left hand side and the $\beta_0(k)$ term, and then the identification need only identify this coefficient for each time step. If implemented this way, there should be no difference between the DMA and the DARMA formulations, since the same information is sought for identification. IV.3. The extra freedom section in computation timing again applies except for item 3 which refers to time-invariant cases. Below we give methods for improvements involving use of adaptive control when the variation of the coefficients is assumed sufficiently slow. IV.4. Under-specification is again possible to guarantee feasibility of the desired trajectory, by using a DARMA model with the appropriate number of terms and implementing as described before. And IV.5, convergence to a minimum least squares trajectory when the desired trajectory is infeasible, is again possible. A potential advantage of the DARMA model version is that there will generally be fewer unknowns to identify.

7. A Modified Learning Control Algorithm to Realize the Benefits of Both Learning and Adaptive Control: Both the DMA and DARMA forms of learning control above, assume that the parameters to be identified can change arbitrarily from one time step to the next. In many practical applications, the time varying coefficients in a system difference equation will vary smoothly and relatively slowly as a the time steps progress. As discussed above, adaptive control cannot hope to produce zero tracking error in a time varying system, but it often gives good performance. As long as the number of time steps required for the adaptive controller to adjust to changed coefficients is small compared to the amount of time needed for the coefficients to change significantly, then adaptive control can be very useful in producing good performance -- and this statement applies within a single repetition.

For such cases where there really is a close relationship between the coefficients identified from one time step to the next, it would be desirable to find a way that the learning control law could benefit from the good performance of an adaptive controller. This can easily be done: 1. During the 0th repetition, only the feedback control law is operating to try to obtain the desired tracking. 2. During the 1st repetition, one can difference the data from the present repetition and the previous repetition so that the deterministic disturbance is absent from the equation. Also, since the initial conditions are the same for each repetition, the initial conditions for the differenced equations are all zero. This is important in the DARMA version when the initial conditions needed *before* the start of a repetition may not be known. Then one can apply adaptive control throughout the first repetition to obtain the benefits of on-line identification of the system model. (If significant noise is present in the data, one may wish to do this for more than one repetition and average the results, before proceeding to use of learning control.) 3. Because of the time lag of the adaptive controller in tracking changing coefficients, the adaptive controller will be producing tracking errors, but if the coefficients change slowly compared to the adaptation time, the errors will be an improvement over the feedback control alone in the 0th repetition. The adaptive control signal history for this repetition is stored and used as the initial learning control function which is altered in the learning process as the repetitions progress. Starting with the 3rd repetition we apply learning control in any of the above forms, in order to learn the changes to make to this recorded adaptive control signal history in order to eliminate the adaptation time lag effect and produce zero tracking error.

In the case where there is no deterministic disturbance term, one can improve upon the above procedure. Two cases apply. In some situations, one knows what the initial conditions should be for the difference equation (8), or preferably written as in equation (9) without having taken the differences between repetitions. For example, one may know that the system is at some position and at rest, in which case, if the output is position, one can set all values before time zero to the same initial value and use zero control actions before time zero. When these initial conditions are known, then one can start the adaptive control from step zero on the 0th repetition, allowing one to gain a time advantage of one repetition. In other cases when one does not know the initial conditions appropriate for the yet to be identified DARMA model (the initial conditions appropriate for a differential equation may more easily be known than those for a difference equation, but there is no way in general to convert them without knowing the system) one can start the adaptive controller at time step $k = \bar{n}$, where again $\bar{n}$ is an upper bound for the order of the system.

V. TIME-INVARIANT SYSTEMS WITH A REPETITIVE DISTURBANCE

When the system is time invariant, new possibilities exist to exploit both learning and adaptive control.

1. The Deterministic Moving Average (DMA) Version of Learning Control: In the time invariant case, all of the entries on the diagonal of the P matrix in (4,5) are identical, as are the entries in each subdiagonal. The estimation process should take advantage of this information, and this can be done by repackaging equation (4) as follows. Define

$$\underline{Y} = \left[\delta_1 \underline{y}^T \quad \delta_2 \underline{y}^T \quad \cdots \quad \delta_r \underline{y}^T \right] \tag{11}$$

and denote the time invariant entries in the matrix P in equation (5) according to

$$\left[\delta_r \underline{y}\right] = \begin{bmatrix} H_1 & 0 & \cdots & 0 \\ H_2 & H_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H_p & H_{p-1} & \cdots & H_1 \end{bmatrix} \left[\delta_r \underline{u}\right] \tag{12}$$

Then the set of equations (4) for a sequence of repetitions through repetition r can be rewritten as

$$\underline{Y} = \left[H_1 \quad H_2 \quad \cdots \quad H_p\right] \underline{W} \tag{13}$$

where

$$\underline{W} = \left[U_1 \quad U_2 \quad \cdots \quad U_r\right] \tag{14}$$

$$U_j = \begin{bmatrix} \delta_j u(0) & \delta_j u(1) & \cdots & \delta_j u(p-1) \\ 0 & \delta_j u(0) & \cdots & \delta_j u(p-2) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \delta_j u(0) \end{bmatrix}$$

Equation (13) eventually becomes an overdetermined set of equations. The identification part of the learning control law is then a recursive least squares algorithm to add one column of $\underline{W}$ at a time to the estimates of H_j. As the time steps progress through the jth repetition, one new column of U_j is added at each time step. The one repetition ahead learning control is obtain as before from the identified P.

All of the comments in subsections IV.2 to IV.5 again apply. However, in IV.2 the learning in a wave becomes significantly easier because instead of needing to identify $C(k+1)B(k)$ for all k as the wave progresses, one only needs to know the constant value CB which can be determined at the start. This much knowledge allows one to learn to produce zero tracking error one new time step per repetition (in the deterministic case without noise), skipping time steps needed for identification except at the beginning. Note that one can identify at the start not only CB but also CAB, and once this is accomplished the wave of producing zero tracking error can progress two steps per repetition, and of course one can generalize this to more steps each repetition. Also, item 3 in subsection IV.3 describes using identification results obtained during a given repetition to improve performance during that repetition, and in this time-invariant situation, this becomes more effective.

2. Indirect Learning Control Based on a Time-Invariant DARMA Model: The algorithm is simplified for this case by noting that (9) becomes

$$\begin{aligned} \delta_j y(k+1) &= \alpha_0 \delta_j y(k) + \ldots + \alpha_{n-1} \delta_j y(k-n) + \beta_0 \delta_j u(k) + \ldots + \beta_l \delta_j u(k-l) \\ \delta_j y(k+1) &= \delta_j \phi^T(k) \theta \; ; \; k = 0, 1, \ldots, p-1 \end{aligned} \tag{15}$$

where the α_i and β_i and θ no longer have time arguments. The least squares identification is applied to these equations starting at the beginning of the first repetition, after the 0th repetition data has been used to produce the differences in (15) and eliminate the repetitive disturbance. Initially, we consider that the identification is done as a batch process for each repetition and then the resulting learning control actions are applied to the next repetition, as was necessary in the time-varying case. The number of unknowns in this DARMA formulation will generally be significantly less than in the DMA in the time-invariant case, giving it an advantage unless one chooses to learn in a wave. Learning can progress in a wave in this DARMA formulation, progressing one step per repetition after having learned β_0, or it can progress in a wave giving zero tracking error for two extra steps per repetition once one has identified β_0, β_1, and α_0.

3. A Modified Learning Control Algorithm to Realize the Benefits of Both Learning and Adaptive Control: In the time-invariant case, the computations required for learning control with a DARMA model starting during the first repetition are essentially the same as adaptive control. The batch processing is appropriate when one cannot make the computations in real time and uses item 4 of subsection IV.3 above. If the hardware is capable of making these computations in real time, then one can improve upon the batch process for each repetition described above, and converge faster to zero tracking error. The full length version of this paper develops the same result (10) above for this case of on-line computation and shows that learning in real time must do at least as well as the above batch learning process. By implementing the algorithm this way, one obtains performance that is both adaptive in time and learning in repetitions. The algorithm is essentially the same as indirect adaptive control, but one has paid a one repetition penalty by staring with the 1st repetition instead of the 0th repetition in order to eliminate the repetitive disturbance which adaptive control cannot easily handle. Also, the system keeps restarting with the same initial conditions every *p* time steps, which is not the standard adaptive control situation, but the proof still works if one applies the final identified model from the previous repetition as the starting model in the next repetition. Thus the benefit of quick reaction in adaptive control, and the ability of learning control to eliminate deterministic disturbances have been realized simultaneously.

VI. TIME-INVARIANT SYSTEMS WITHOUT A REPETITIVE DISTURBANCE

When there is no repetitive disturbance, one can start learning during the 0th repetition. Using the DMA model, one can start at time step zero if one knows the initial state of the system. Using the DARMA model, if one knows appropriate initial conditions to use for the higher order difference equation model, then one again can start with the first time step of the 0th repetition. Otherwise, one must wait until the time step in which the difference equation in (15) no longer contains unknown and unmeasured values. It is in time invariant linear systems without disturbances that indirect adaptive control gives conclusive convergence results saying that in the limit as time tends to infinity an adaptive controller will produce zero tracking error. The appropriate learning control is that described in V.3, but altered to start during the 0th repetition as soon as possible without the need to difference two successive repetitions. This learning control law for a time invariant problem without a repetitive disturbanceis is only one step removed from the associated indirect adaptive control. It differs in only one respect -- the fact that the system keeps restarting every *p* steps from the same initial conditions.

Thus, with this learning control law the bridge is now complete, connecting pure indirect adaptive control (designed to give guaranteed convergence to zero tracking error in time-invariant systems without any deterministic exogenous driving terms) to indirect learning control (capable of handling time varying systems, that are restarted every *p* time steps, and that have an unknown deterministic disturbances).

SUMMARY

In summary, a rather complete matrix of the ways in which indirect adaptive control ideas can be applied to the learning control problem has been presented. The methods differ based on the type of system model considered (DMA or DARMA) and on whether the system is time-varying or time-invariant, and whether there is a deterministic disturbance. The learning control methods guarantee convergence to zero tracking error with a deterministic model in situations in which adaptive control could not obtain zero error. By appropriate design of the learning control algorithms, the advantages of quicker response that adaptive control offers have been incorporated into the learning process to the extent possible for the type of system model used. Establishing convergence requires some extension of the proof of convergence of indirect adaptive control theory, and appropriate application of the ideas in the learning control situation.

REFERENCES

1. Middleton, R. H., Goodwin, G. C., and Longman, R. W., "A Method for Improving the Dynamic Accuracy of a Robot Performing a Repetitive Task," *International Journal of Robotics Research*, Vol. 8, No. 5, October 1989, pp.67-74. Also available as University of Newcastle, Newcastle, Australia, Department of Electrical Engineering Technical Report EE8546, 1985.

2. Harokopos, E. G., and Longman, R. W., "A Learning Controller for Tracking Arbitrary Inputs in Robot Motions," *Proceedings of the Tenth IASTED International Symposium on Robotics and Automation*, Lugano, Switzerland, June 29-July 2, 1987.

3. Phan, M., and Longman, R. W., "A Mathematical Theory of Learning Control for Linear Discrete Multivariable Systems," *Proceedings of the AIAA/AAS Astrodynamics Conference*, Minneapolis, Minnesota, August 1988, pp. 740-746.

4. Phan, M., and Longman, R. W., "Liapunov Based Model Reference Learning Control," *Proceedings of the Twenty-Sixth Annual Allerton Conference on Communication, Control and Computing*, The University of Illinois, Allerton House, Monticello, Illinois, September 1988, pp. 927-936.

5. Phan, M., and Longman, R. W., "Indirect Learning Control with Guaranteed Stability," *Proceedings of the 1989 Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, MD, 1989, pp.125-131.

6. Longman, R. W., Beigi, H. S. M., and Li, C. J., "Learning Control by Numerical Optimization Methods," *Modeling and Simulation*, Instrument Society of America, Vol. 20, Part 5, 1989, pp. 1877-1882.

7. Phan, M., and Longman, R. W., "Learning System Identification," *Modeling and Simulation*, Instrument Society of America, Vol. 20, Part 5, 1989, pp. 1857-1864.

8. Phan, M., and Longman, R. W., "A Model Reference Adaptive Control Method for a Class of Linear Discrete Systems," *Modeling and Simulation*, Instrument Society of America, Vol. 20, Part 5, 1989, pp. 1865-1869.

9. Phan, M., *A Mathematical Theory of Learning Control*, Ph.D. Dissertation, Columbia University, New York, N.Y., May 1989.

10. Phan, M., Wei, Y., Horta, L. G., Juang, J.-N., and Longman, R. W., "Learning Control for Slewing of a Flexible Panel," *Dynamics and Control of Large Structures*, Proceedings of the Seventh VPI&SU/AIAA Symposium, L. Meirovitch, Editor, May 1989, pp. 39-52.

11. Juang, J. N., Horta, L., and Longman, R. W., "System Identification in the Repetition Domain," Proceedings of the 1989 AAS/AIAA Astrodynamics Conference, Stowe, VT, August 1989, *Advances in the Astronautical Sciences*, American Astronautical Society, Vol. 71, 1990. Also, Juang, J.-N., Horta, L. G., and Longman, R. W., "Input/Output System Identification: Learning from Repeated Experiments," Chapter 5 of *Mechanics and Control of Large Flexible Structures*, J. L. Junkins, Editor, Progress in Astronautics and Aeronautics, Vol.129, American Institute of Aeronautics and Astronautics, Washington, DC, 1990, pp.87-99.

12. Phan, M., Longman, R. W., and Juang, J-N., "Indirect Repetitive Control for Linear Discrete Multivariable Systems," *Proceedings of the 27th Annual Allerton Conference on Communication, Control, and Computing*, The University of Illinois, Allerton House, Monticello, Illinois, September 1989, pp.867-876.

13. Phan, M., Juang, J.-N., and Longman, R. W., "Recent Developments in Learning Control and System Identification for Robots and Structures," *Dynamics of Flexible Structures in Space*, Proceedings of the First International Conference, Cranfield, England, May 1990, C. L. Kirk, and J. L. Junkins, Editors, Computational Mechanics Publications, Southampton, Boston, and Springer-Verlag, New York, co-publishers, 1990, pp. 321-334.

14. Longman, R. W., and Chang, C.-K., "Learning Control for Minimizing a Quadratic Cost during Repetitions of a Task," *AIAA/AAS Astrodynamics Conference, A Collection of Technical Papers*, Part 2, Portland, Oregon, August 1990, pp. 530-536.

15. Longman, R. W., and Solcz, E. J., "Small Gain Robustness Issues in the P-Integrator Repetitive Controller," *AIAA/AAS Astrodynamics Conference, A Collection of Technical Papers*, Part 2, Portland, Oregon, August 1990, pp. 537-551.

16. Solcz, E. J., and Longman, R. W., "Disturbance Rejection in Repetitive Controllers," *Advances in the Astronautical Sciences*, American Astronautical Society, Vol. 76, 1992, Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado,

August 1991.
17. Chang, C.-K., Longman, R. W., and Phan, M., "Techniques for Improving Transients in Learning Control Systems," *Advances in the Astronautical Sciences*, American Astronautical Society, Vol. 76, 1992, Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado, August 1991.
18. Lee, S. C., Longman, R. W., and Phan, M. "Linear Decentralized Learning Control," *Advances in the Astronautical Sciences*, American Astronautical Society, Vol. 76, 1992, Proceedings of the AAS/AIAA Astrodynamics Specialist Conference, Durango, Colorado, August 1991.
19. Beigi, H. S. M., Li, C. J., and Longman, R. W., "Learning Control Based on Generalized Secant Method and other Numerical Optimization Methods," 1991 ASME Winter Annual Meeting, Atlanta, Georgia, *Symposium on Sensors, Controls, and Quality Issues in Manufacturing*, PED-Vol. 55, December 1991, pp. 163-175.
20. Longman, R. W., Lee, S. C., and Phan, M., "Indirect Decentralized Learning Control," *Advances in the Astronautical Sciences*, American Astronautical Society, to appear, Proceedings of the 1992 AAS/AIAA Spaceflight Mechanics Meeting, Colorado Springs, February, 1992.
21. Solcz, E. J., and Longman, R. W., "Robustness of Repetitive Controllers to Frequency Uncertainties," *Proceedings of the 1992 Conference on Information Sciences and Systems*, Department of Electrical Engineering, Princeton University, March 1992, to appear.
22. Solcz, E. J., and Longman, R. W., "Repetitive Control in the Presence of Multiple Periodic Disturbances," *Proceedings of the 1992 AIAA/AAS Astrodynamics Specialist Conference*, an AIAA publication, Hilton Head, South Carolina, August 1992.
23. Chang, C. K., Longman, R. W., and Phan, M., "Discrete Time Learning Control in Nonlinear Systems," *Proceedings of the 1992 AIAA/AAS Astrodynamics Specialist Conference*, an AIAA publication, Hilton Head, South Carolina, August 1992.
24. Longman, R. W., Phan, M., and Juang, J.-N., "An Overview of a Sequence of Research Developments in Learning and Repetitive Control," *Proceedings of the First International Conference on Motion and Vibration Control (MOVIC)*, Yokohama, Japan, September 1992, to appear.
25. Arimoto, S., "Introduction: Brief History of Development of Learning Control in Robotics," appearing in *Supplementary Material, Tutorial on Learning Control*, Organizers: R. W. Longman, S. Arimoto, Additional Lecturers: M. Phan, and J.-N. Juang, 1991 IEEE Conference on Intelligent Control, Arlington, VA, August 1991.
26. Arimoto, S., Kawamura, S., and Miyazaki, F., "Bettering Operation of Robots by Learning," *Journal of Robotic Systems*, Vol.1, No. 2, 1984, pp. 123-140.
27. Craig, J. J., "Adaptive Control of Manipulators Through Repeated Trials," *Proceedings of the American Control Conference*, San Diego, 1984, pp. 1566-1573.
28. Casalino, G., and Bartolini, G., "A Learning Procedure for the Control of Movements of Robotic Manipulators," *IASTED Symposium on Robotics and Automation*, Amsterdam, 1984, pp. 108-111.
29. Uchiyama, M., "Formulation of High-Speed Motion Pattern of a Mechanical Arm by Trial," (in Japanese), *Transactions of the Society for Instrumentation and Control Engineers*, Vol. 14, 1978, pp. 706-712.
30. G. C. Goodwin and K. S. Sin, *Adaptive Filtering Prediction and Control*, Prentice Hall, 1983.

Active Control of Machining Processes Using Image Reconstruction and Processing

D. DeVoe, L. Knox, and G. Zhang

Mechanical Engineering Department and Systems Research Center
University of Maryland at College Park
College Park, MD 20742

Abstract

Computer Integrated Manufacturing (CIM) systems have emerged in response to the need for high product quality and productivity. Implementation of computer-based untended machining has been the focus of current research in manufacturing systems. This paper presents the development of a computer-based vision system which performs in-line monitoring of surface roughness during a machining process. Acting as the sensing device, a monochrome CCD camera is used to capture a gray-scale image from an area being machined. The developed image manipulation software reconstructs the three-dimensional surface topography from the gray-scale image for the purpose of controlling the machining accuracy. Experimental results from four aluminum samples with different surface topographical characteristics are presented to demonstrate the system's capability of providing a quality control check during the machining process.

1. Introduction

Producing the proper surface finish is an important part of any machining process. In order to ensure that machined parts have been manufactured with the desired surface roughness, a sample of the parts is typically examined after machining. The primary disadvantage of this post-machining inspection process is that it does not allow for corrections to be made during machining. Additionally, it becomes tedious to sort defective parts, since each part must be examined individually after being machined. The need for improved flexibility, productivity, and product quality in a modern manufacturing environment points to the need for high-speed in-process surface roughness measurement systems. In-process inspection of surface roughness during a machining process can provide a quality control check on the process, indicating if the surface finish of the machined part is within some predefined tolerance, without disrupting the production process. In-process inspection systems represent the key to successfully controlling product surface quality in a sensor-based manufacturing environment.

Regarding in-process inspection, traditional methods of surface roughness assessment have the disadvantage of requiring direct contact with the surface, as is the case with the common stylus-type devices. Such contact-based techniques are less than ideal for in-process inspection of surface roughness for several reasons:

- the measurement speed is limited due to the need for physical contact with the surface.
- non-continuous surfaces are extremely difficult to inspect, as the stylus tip must be navigated around any discontinuities in the surface.
- surface damage may occur due to wear between the stylus tip and the surface under inspection.
- the apparatus must be carefully isolated from vibrations, which can become difficult in a machining environment.
- roughness values are determined using a limited number of single-line traces, which may not be representative the entire surface.

- the surface must be free of oil, dirt, and other contaminants -- in an actual machining environment, the surface must be cleaned, measured, and re-lubricated, which is expensive to implement as a high speed in-line process.

Recent research efforts in the manufacturing community have been focused on the development of improved, high precision machines, equipment and processes through advances in dimensional measurement techniques, sensors to monitor and control precision machines, and physical models of precision machining processes. The use of in-process surface roughness assessment represents one such area of interest. Several non-contact methods have been suggested as alternative surface roughness measurement techniques more suited for in-process inspection than traditional methods. For instance, laser-based profilometry systems and linear diode arrays can achieve faster measurement speeds (Jansson, 1984) than stylus methods by measuring the intensity of light reflected from the surface, but these methods suffer from an inability to characterize the entire surface topography. The ideal measurement system should combine the speed of laser-based systems, the accuracy of contact profiling techniques, and the ability to characterize a 2-dimensional region of the surface found in area-based topographic techniques such as capacitance or ultrasonic methods. This paper presents an experimental study of an area-based surface roughness assessment technique using image processing, which addresses each of these concerns.

2. Description of the Image-Processing Based Method

The roughness measurement system proposed here applies the light-scattering theory developed by Beckman (1963), which shows that the intensity field of light scattered from a rough surface can be described as a function of the surface topography. Using this insight, it should be possible to reconstruct an object's topographic (i.e. amplitude and wavelength) information, and thus surface roughness, by analyzing the pattern of light scattered from its surface. Beckman's theory applies to a single wavelength light source, but it may be extended for use with a white light source by superposition of the individual wavelengths of the white light. While other roughness measurement systems have applied scattering theory through analysis of reflected intensity over a range of incident and collection angles, the method presented here considers fixed incident and collection angles. This latter system promises to be simpler and less expensive to implement as an active roughness monitoring process.

The photo-optical measurement method proposed here uses the apparatus shown in Figure 1. Here a sample is illuminated by a light source, and a CCD camera using a high magnification lens system provides a video signal which a frame grabber converts into an 8-bit digital image in real-time (30 frames per second). This image is then sent to a microcomputer for processing. The computer examines the light scattering pattern in the image, and calculates the roughness parameter of the surface from the image's gray-level histogram. The R_a value for the surface is then determined through the use of a correlation curve which uniquely relates a range of the roughness parameter to a range of R_a values. The resulting R_a value is either displayed on a video monitor for observation or used as feedback to the machining process.

The proposed apparatus avoids the pitfalls encountered by other methods of in-process roughness inspection. Since there is no physical contact with the surface, drawbacks such as low measurement speed, the need for navigation of non-continuous surfaces, and potential surface damage no longer present a problem. Vibration isolation is not difficult as the camera may be mounted relatively far from the surface under inspection. Since this technique is an area-based measurement method, it is capable of determining roughness values from a 2-dimensional region of the surface, and thus provides data more representative of the entire surface than single-line tracing methods.

Work on this topic has been done by Luk et al. (1989) in which a machine vision system was used to determine the surface roughness of several steel, copper, and brass samples. In this work, a camera and frame grabber were used to digitize an area of each sample, and the optical roughness parameter (R = std.dev./RMS) was calculated from the gray-scale images. A relationship was developed between known roughness values and the optical roughness parameter, and this relationship was used to test the repeatability of the optical method by comparing roughness values derived using the optical technique and a stylus device. It was found that the optical method produced more consistent readings than the stylus method when multiple measurements were made on a single sample, demonstrating that a computer vision system can be used to repeatably measure surface roughness.

The primary focus of this work has been in determining how to make the vision system capable of measuring surface roughness with sufficient accuracy. Ideally, the system precision would be in the same range as that of standard stylus measurement techniques. While the work by Luk et al. demonstrates the repeatability of vision-based roughness measurement systems, the potential accuracy of such systems remains to be seen. In order to make such a system viable for use in an automated industrial quality control system, the system must be capable of delivering high accuracy surface roughness measurements. We believe that the accuracy of the system is especially sensitive to the lighting conditions used to illuminate the surface, and that determining the ideal lighting conditions will result in improved system accuracy. Additionally, it must be determined which parameters of the digitized image can be best correlated to the surface roughness in order to provide acceptable accuracy and sensitivity to changes in R_a.

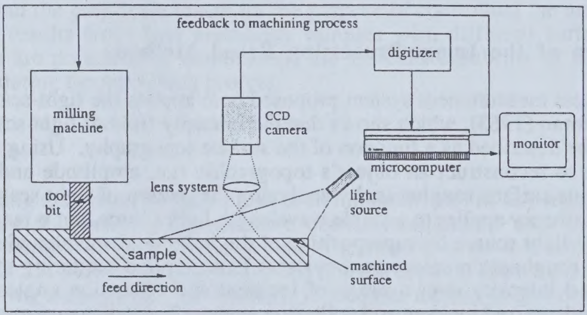


Figure 1 – Ideal optical apparatus

3. Experimental Study

The experimental study began with the preparation of samples of machined surfaces as illustrated in Figure 2. Four samples of aluminum, labeled A through D, were prepared under four different machining parameter settings to produce different surface roughnesses.

Each sample was photographed using a Nikon camera using high resolution black and white Kodak film. A high magnification lens system, capable of filling a standard 3 1/2" x 5" photograph with a 2.0cm x 1.4 cm image of the sample, was employed. Direct lighting was achieved using a 3.6W white light bulb to produce three sets of images of each sample. The lighting arrangement for each set is depicted in Figure 3. A set of photographs was also made under diffuse conditions, produced using a 150W light projected from behind the samples onto a white, granular mat designed to produce diffuse lighting conditions. Baseline conditions between different samples for a given lighting setup were maintained by simply swapping the samples without altering either the lighting or camera setup.

An additional photograph was made of a sheet of precision graph paper placed at the same focal point used for the sample photographs. This photograph was used to determine the scale of the samples. The photographs produced under diffuse lighting conditions are shown in Figure 4.

Each of the pictures resulting from the photo-optical apparatus were scanned into a Macintosh II microcomputer using an 8-bit (256 gray level) scanner, and the resulting gray-scale images were loaded into an image manipulation program for processing (*NIH Image*[*] running on a Macintosh II microcomputer was used for this purpose). Using a variety of histogram analysis tools for *NIH Image*, several macros were written to determine three imageparameters: standard deviation (SD), arithmetical average deviation (AAD), and RMS values based on the brightness level histogram (typified by the histogram in Figure 5) over a 12.5 mm^2 region of each image. These parameters are defined as follows:

$$SD = \sqrt{\frac{1}{n-1}\sum_{i=0}^{255} F_i (x_i - \bar{x})^2} \qquad RMS = \sqrt{\frac{1}{n}\sum_{i=0}^{255} F_i^2} \qquad AAD = \frac{1}{n}\sum_{i=1}^{n} |x_i - \bar{x}|$$

where n = # of pixels in the image, $\bar{x}$ = mean gray (brightness) level of all pixels in the image, x_i = gray (brightness) level of the i^{th} pixel ($0 \le i \le 255$), and F_i = frequency count of pixels in the image at brightness level x_i. The parameter values calculated for each sample are listed in Table 1.

Table 1 - Optical and Stylus Parameters

	(NIST)	*(Optically*	*Derived*	*Parameters)*
Sample	**R_a**	**AAD**	**RMS**	**SD**
A	2.9	18.109	134.909	12.872
B	0.9	85.578	96.362	45.808
C	4.9	33.037	119.993	28.453
D	4.0	51.892	107.737	33.639

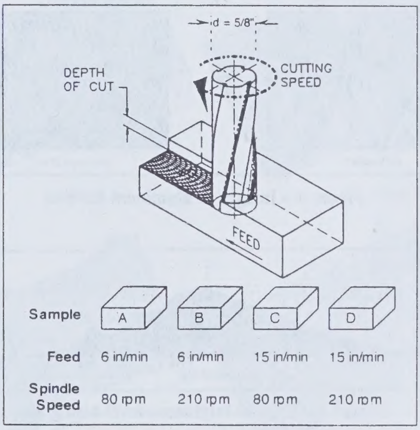


Figure 2 – Machining parameter settings used to produce experimental samples

With the image parameters determined, calibration data was required to produce the desired correlation curves. Facilities at the Ceramics Division of the National Institute of Standards and Technology (NIST) were used to measure the sample surface roughnesses. A Perthometer stylus device was used to calculate the average roughness index, R_a, for each sample averaged over multiple measurement trials. In addition to these stylus measurements, a

---

[*] NIH Image is a public domain image processing and analysis tool for the Macintosh II microcomputer written by Wayne Rasband at the National Institutes of Health

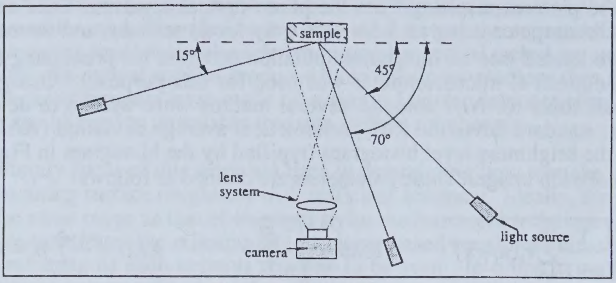


Figure 3 – Lighting system

series of 151 parallel traces were taken on the samples using a custom stylus-based apparatus developed at NIST. This device was designed for the purpose of producing a 3-dimensional reconstruction of the surface topography over a region of a surface. Figure 6 presents surface profiles traced from each of the four samples. The average R_a values are listed in Table 1.

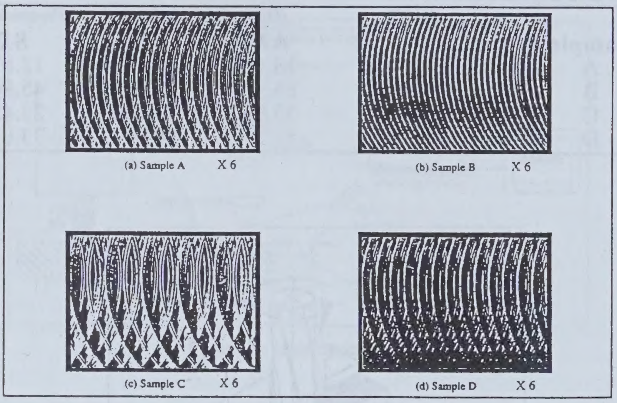

(a) Sample A X 6
(b) Sample B X 6
(c) Sample C X 6
(d) Sample D X 6

Figure 4 – Images of aluminum samples

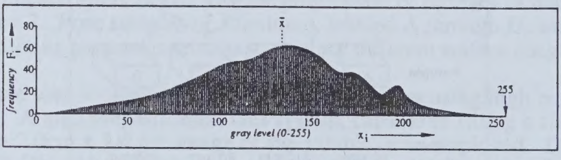
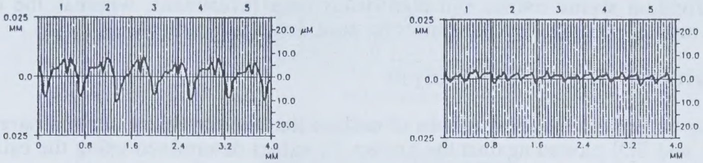


Figure 5 – Typical brightness-level histogram

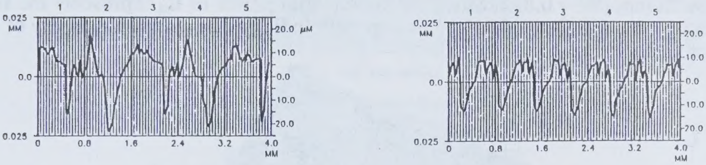
4. Discussion of Results

4.1 Topographical Visualization

Numerical data obtained from both the stylus measurement and optical inspection methods are listed in Table 1. However, variations of both R_a and optical parameter values were observed across individual samples. These variations are due to the nature of the machining process employed to produce the samples. Therefore, the R_a data listed in Table 1 represents the mean level of R_a for the given sample. The variation of R_a across each sample can be observed qualitatively in the 3-dimensional reconstructions of the sample surfaces produced using *NIH Image*. These images, shown in Figure 7, were created by plotting the brightness level of the scanned image along the z-axis, with the sample's spatial coordinates in the x-y plane.



(a) Sample A, $\overline{R}_a = 2.9\ \mu m$ (b) Sample B, $\overline{R}_a = 0.9\ \mu m$



(c) Sample C, $\overline{R}_a = 4.9\ \mu m$ (d) Sample D, $\overline{R}_a = 4.0\ \mu m$

Figure 6 – Surface profiles of each sample

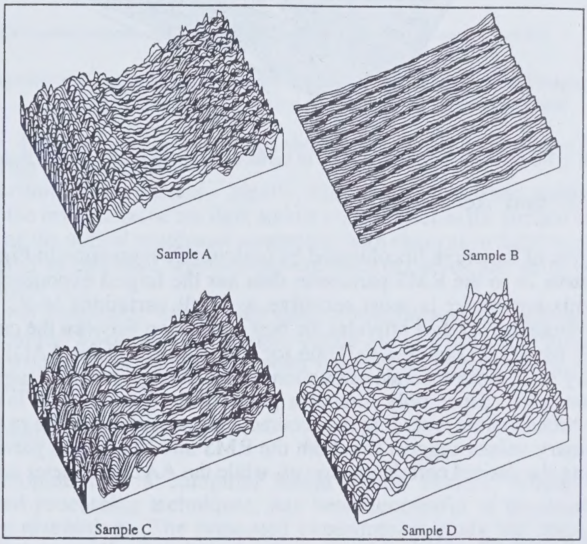


Figure 7 – 3-dimensional reconstructions of surface topography using Image software

Examining these four topographic reconstructions, it is evident that the irregularities displayed in the topographic reconstruction from sample C, which has the largest R_a value, are much larger than those displayed in the topographic reconstructions from sample B, which has the smallest R_a value. This observation confirms that the optical inspection method has the capability to distinguish between different patterns of surface roughness. Figure 8 is the surface topography reconstructed from the surface roughness profilometer measurements.

Similarity in appearance can be observed between the two methods of reconstruction. This also confirms the validity of applying the optical inspection method to surface roughness assessment. It should be noted that the profilometer-based topographic reconstruction required over 150 individual stylus traces, and significant time investment, whereas the optical measurement method produced similar results for considerably less effort.

4.2 Logarithmic Regression Analysis

Figures 9a through 9c show graphs of each of the three optically-derived parameters (RMS, AAD, and SD) plotted against the known R_a values determined using the calibrated stylus measurement system at NIST. A logarithmic curve was found to achieve a good fit through the data, and these curves are shown in each of the graphs. For example, the logarithmic fit to the SD data results in the relationship $SD = 14.8(R_a^{0.659})$, with a good correlation coefficient ($R^2 = 0.852$. Since the exponential power of R_a represents the slope of the curve in log-space, a higher value for this exponent indicates increased sensitivity of the SD parameter to changes in R_a.

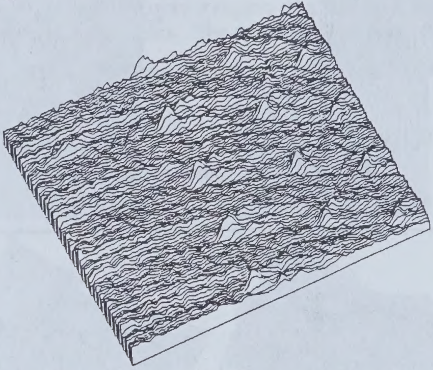


Figure 8 – 3-dimensional reconstructions of surface topography using stylus instrument

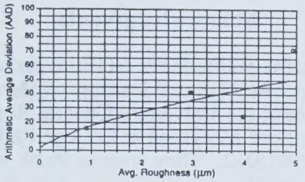
4.3 Sensitivity and Accuracy

Examination of the curve fits obtained by logarithmic regression in Figs. 9a through 9c shows that the curve fit to the RMS parameter data has the largest exponential value for R_a, indicating that this parameter is most sensitive to small variations in R_a. However, the logarithmic fit through the SD data provides the best correlation between the curve and data (an R^2 value of 0.852 for SD, as opposed to 0.666 for RMS and 0.637 for AAD), suggesting that this parameter may yield more accurate estimations of the average surface roughness than either of the other parameters. Since only four data points were used for this investigation, it is difficult to make broad claims regarding the accuracy and sensitivity of the optical parameters, but these preliminary values suggest that both the RMS and SD optical parameters are most useful in producing the desired correlation curves, while the AAD parameter is less than ideal.

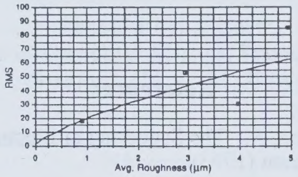
4.4 Further Research

Clearly, more data needs to be collected in order to develop accurate correlation curves which are reasonably sensitive to variations in R_a. As well, the following topics deserve special attention for future research:

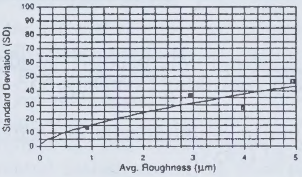
- *determination of ideal lighting conditions* - the intensity, incident angle, and proximity of the light source used to illuminate the surface were observed to have a strong effect on the optical parameters of the surface. The sensitivity of the optical parameters to, and the system



(a) Arithmetic average deviation AAD = $19.3(Ra^{0.693})$, and a correlation coefficient of $R^2 = 0.595$



(b) RMS, RMS = $16.5(Ra^{0.732})$, and a correlation coefficient of $R^2 = 0.666$



(c) Standard deviation, SD = $14.8(Ra^{0.659})$, and a correlation coefficient of $R^2 = 0.852$

Figure 9 – Relationship between values of three parameters determined using optical inspection method and average roughness values determined using stylus instrument

accuracy under, different lighting conditions should be examined in order to determine which conditions provide the best results.

- *surface orientation independence* - ideally, the measured surface roughness should be independent of the orientation of the light source with respect to the surface under inspection. The variability of the optical roughness parameters with changes in lighting orientation needs to be investigated to establish whether the correlation curves are valid only for a given lighting orientation.
- *operable range of roughness values* - the surface roughnesses of the four samples in this investigation covered a wide range between 0.9µm and 4.9 µm. It is plausible that better results may have come from studying a smaller range of roughness values, or a range with a smaller upper limit.

5. Conclusions

The development of a computer based vision system, which utilizes image reconstruction and processing techniques, has been successful in monitoring the surface roughness during machining. The presented experimental study has shown not only the feasibility in real-time application, but also the potential of integrating sensing devices and image processing software into manufacturing systems for the realization of production automation. As indicated in the paper, further investigation is needed to examine the accuracy of the system calibration and sensitivity of the surface characterization indices on the detection of abnormalities in the machining process.

Acknowledgments

The authors wish to express gratitude for facilities and assistance provided by Drs. L. Ives, S. Jahanmir, E. Whitenton of the National Institute of Standards and Technology. The authors acknowledge the support of the University General Research Board, and the Systems Research Center at the University of Maryland at College Park under Engineering Research Centers Program: NSFD CDF 8803012. They wish to thank Ms. Linhna Duong and Mr. Jei Wen for their assistance.

References

1. Beckmann P., Spizzichino, A., The Scattering of Electromagnetic Waves From Rough Surfaces, Oxford:Pergamon (1963).

2. Bolz R., Designing for Production ASME Handbook: Metals Engineering Design, 277-83 (1953).

3. Booker G., Scanning Electron Microscopy Modern Diffraction and Imaging Techniques, North Holland Publishing Company, 553-97 (1970).

4. Broadston J., Surface-Finish Requirements in Design ASME Handbook: Metals Engineering Design, 277-83 (1953).

5. Jansson D. G., Bourke J. M., *High-speed surface roughness measurement*, J. Eng. Ind. 106, 34-9 (1984).

6. Lyman T. (ed.), ASM Metals Handbook: Machining, American Society for Metals, 8th ed., 3, 169-93 (1967).

7. Luk F., Huynh V., North W., *Measurement of Surface Roughness by a Machine Vision System*, Journal of Physics E: Scientific Instruments 22, 977-80 (1989).

8. Thomas T., Rough Surfaces, Longman Group Ltd., Harlow, Essex UK (1982).

9. Vorburger T., Raja J., Surface Finish Metrology, NISTIR-4088, U.S. Dept. of Commerce (1990).

Nomenclature

R_a = Roughness average parameter
R = Optical roughness parameter (=SD/RMS)
SD = Standard deviation
AAD = Arithmetic average deviation
RMS = Root mean square value
n = Number of pixels in an image
$\overline{x}$ = Mean brightness level of all pixels in an image
x_i = Gray (brightness) level ($0 \leq i \leq 255$)
F_i = Frequency count of pixels in an image at brightness level x_i
R^2= Correlation Coefficient

Reasoning About Mechanical Assembly

Randall H. Wilson
rwilson@cs.stanford.edu

Jean-Claude Latombe
latombe@cs.stanford.edu

Robotics Laboratory
Department of Computer Science, Stanford University
Stanford, CA 94305, USA

Abstract

In which order can a product be assembled or disassembled? How many hands are required? How many degrees of freedom? What parts should be withdrawn to allow the removal of a specified subassembly? To answer such questions automatically, important theoretical issues in geometric reasoning must be addressed. This paper investigates the planning of assembly algorithms specifying (dis)assembly operations on the components of a product and the ordering of these operations. It also presents measures to evaluate the complexity of these algorithms and techniques to estimate the inherent complexity of a product. The central concept underlying these planning and complexity evaluation techniques is that of a "non-directional blocking graph", a qualitative representation of the internal structure of an assembly product. This representation is obtained by identifying physical criticalities where interferences among parts change. It is generated from an input geometric description of the product. The main application considered in the paper is the creation of smart environments to help designers create products that are easier to manufacture and service. Other possible applications include planning for rapid prototyping and autonomous robots.

Acknowledgments: This research was funded by DARPA contract N00014-88-K-0620 (Office of Naval Research), the Stanford Integrated Manufacturing Association (SIMA), and Digital Equipment Corporation. The authors also thank Leo Guibas, Rajeev Motwani, and Achim Schweikard for their encouragement and comments.

1. INTRODUCTION

Reasoning about mechanical assembly (and disassembly) is an important research topic which has attracted interest from researchers in both artificial intelligence [38] and robotics [27]. In which order can a product be assembled or disassembled? How many hands are required? How many degrees of freedom? What parts should be removed from the assembled product to allow replacement of a specified subassembly? To automatically answer questions like these, interesting theoretical issues in geometric reasoning must be addressed and computational techniques of broad interest must be developed. This investigation will also benefit many applications. The one considered in this paper is the creation of smart interactive environments to help designers create products that are easier to manufacture and service. Other important applications include planning for rapid prototyping and autonomous robots.

Given a mechanical assembly product (such as a toaster, an automobile, or a plane), an *assembly algorithm* specifies assembly and/or disassembly operations on its components and the ordering of these operations. The manufacturing, maintenance, and repair procedures for the product are all instances of assembly algorithms. These algorithms may be executed by humans (where the algorithm takes the form of an instruction sheet), robots (here expressed as a robot program), or specific machines (here an input to the design of these machines). Assembly algorithms are the key link between design and manufacturing/servicing. If we can measure the cost of executing an assembly algorithm, then the product's "inherent complexity", with respect to manufacturing and servicing, is given by the lower bound cost of all the assembly algorithms that are possible for this product.

Here we investigate the following two related problems: (1) the automatic generation of assembly algorithms from CAD data, and (2) the characterization of the complexity of assembly designs. Our main goal is to provide efficient computational support for the "concurrent engineering" approach to design, in which constraints arising from manufacturing and servicing are taken into account at design time [9]. As products are designed with more parts of various sorts (e.g., machined, composite, electrical, electronic) densely packed to provide more functions per cubic inch, the need for powerful assembly support tools will increase dramatically.

A typical CAD model of an assembly describes the geometry of the parts composing the assembly and the spatial relations among these parts. It is appropriate for graphic rendering and some man-machine interaction, but it does not directly provide the information that is needed to easily plan assembly algorithms. Indeed, to synthesize such algorithms one must first analyze how the various components in an assembly constrain their respective motions. This suggests that the CAD model be converted into another representation making these constraints explicit. We propose one such representation, called the *non-directional blocking graph*, or NDBG. It derives from the observation that infinite families of motions can be partitioned into finite collections of subsets such that the interferences among the parts are constant over every subset. Once computed, the NDBG can be exploited for a variety of purposes, including the efficient generation of assembly algorithms.

There are often many algorithms to assemble or disassemble a product, and we may not want to produce all. To deal with this issue, we propose to measure the complexity of an algorithm along various axes, e.g., the number of hands it requires, the longest sequence of operations that cannot be parallelized, the total number of elementary motions. Along each axis, we define the algorithmic complexity of a product as the lower bound on the complexity of all assembly algorithms for this product. This definition immediately yields various classes of products, for instance the class of products that can be (dis)assembled with single translations only. The complexity of a product and its class provide concise pertinent information to feed back to the designers during design; it can also be used as a reference to select the most interesting assembly sequences. We will see that the NDBG is directly exploitable to estimate various complexity measures of a design.

The NDBG is a qualitative representation of the internal structure of an assembly product, which is obtained by identifying physical criticalities where interferences among parts change. Constructing the NDBG of an assembly design is a precomputation step allowing subsequent computations (synthesis of assembly algorithms, evaluation of complexity measures) to be performed in efficient query time. We will describe several types of NDBGs (for different families of motions) whose computation takes polynomial time in the size of the CAD input (number and complexity of parts). This computation can be done from a full assembly model, or incrementally, while the assembly is being designed. The exploitation of the NDBGs can be aimed at answering specific requests from the designers (e.g., how many parts must be removed from the assembled product before a specified subassembly can itself be extracted?) and/or posting warnings to the designers (e.g., when the product changes from one complexity class to another).

Notation: The following notation will be used throughout the paper:
- n: number of parts (polygons or polyhedra) in an assembly A,
- v: total number of vertices of the parts in A,
- r: number of pairs of parts in contact in A,
- c: number of contacts (edge-edge or plane-plane) in A,
- k: total number of vertices in the convex hulls of the c contacts in A.

All other notations are defined and used locally.

2. RELATED WORK

The automatic planning of assembly and disassembly operations has attracted the interest of AI researchers for a long time. The classical blocks world can be seen as a primitive assembly planning domain [15, 38]. Moreover, some AI planners have considered more complex domains. For instance, NOAH [42] was originally aimed at supplying instructions to a human apprentice to repair an air compressor, including disassembly and assembly plans. However, because they are usually interested in more general planning issues than just assembly planning, virtually all AI planners make use of a very abstract geometric description of the objects and their relations expressed in logical notation, e.g. `ON(A,B)`. Additional geometric knowledge is implicitly coded in the operators representing the actions that can be executed (e.g., stacking a block on top of another). A noticeable exception is BUILD [13], which includes a simple treatment of such notions as stability and friction. The interest of AI in general planning is still very high [35].

The geometric approach to assembly planning originated in robotics with the work reported in [29] (AUTOPASS), [31] (LAMA), and [45]. It is more limited in scope than traditional AI planning and focuses specifically on issues raised by the manipulation of physical objects. It has motivated various research in basic path planning, motion planning with uncertainty, manipulation planning with movable objects, and grasp planning [27]. However, the high complexity of assembly planning when viewed as a general motion planning problem has led researchers to turn their attention toward a simpler subproblem known as *assembly sequence planning*, or simply *assembly sequencing* [1, 22]. In this problem, only the constraints (mainly geometric ones) arising from the assembly itself are considered; the manipulation system (e.g., the robots) executing this plan is simply ignored by assuming that the parts composing the assembly are flee-flying objects.

The early assembly sequencers were mainly sequence editors. Geometric reasoning was supplied by a human who answered questions asked by the computer systems; the assembly sequences were inferred from the answers to these questions [7, 10]. Automated geometric reasoning was later added to answer these questions automatically [5, 19, 20, 28, 30, 47, 51]. This development resulted in generate-and-test assembly sequencers, with a module guessing candidate sequences and generating questions to check their feasibility, and geometric reasoning modules answering these questions. This approach tends to repeat the same geometric computations many times. Mechanisms for saving and reusing previous computations, such as the "precedence expressions" [47], have been proposed to overcome this drawback, but with limited success. In practice, the generate-and-test paradigm remains relatively inefficient (it takes time exponential in the number of parts) and is applicable only to assemblies with few parts. The NDBG avoids this combinatorial trap by generating only valid operations; it allows polynomial assembly sequencing.

Originally, the research in assembly planning was aimed at assisting process planning in order to reduce delays between design and manufacturing, and eventually produce better plans [10]. This goal is still valuable, especially for rapid prototyping and even mass production. Recently, however, the interest has shifted toward generating assembly sequences to evaluate assembly designs and help designers create products that are easier to manufacture [44, 48]. In this new context, automated geometric reasoning and computational efficiency of assembly planning are critical issues that must be thoroughly explored. The synthesis of pertinent information to feed back to designers is another important issue. The concept of the algorithmic complexity of an assembly design presented in this paper directly addresses this issue. It derives in part from informal complexity measures currently in use in several companies (e.g., see [6]).

The field of computational geometry has also explored issues relevant to assembly planning, for example, set separation problems [46]. Given a 2D polygonal assembly A, the problem of deciding whether there is a direction d and a subassembly $S \subset A$ such that a translation along d separates S from the rest of A is addressed in [4]. An algorithm to construct a sequence of translations separating two polygonal parts is given in [39]. Several techniques presented in this paper have been influenced by the work in computational geometry.

The construction of an NDBG is based on the identification of physical criticalities to decompose a continuous set into a finite number of regions that are treated as single entities. This approach relates to the general interests of qualitative physics [11] and, more specifically, qualitative kinematics [14, 24], which studies the internal motions of parts in an operational device. It yields more meaningful decompositions than blind discretizations not based on any sort of criticality (discontinuity, singularity, or event). The notion of an "aspect graph" used in computer vision also has the same qualitative flavor as the NDBG. The aspect graph of an object describes the appearance of the object from all possible points of view. Aspect graphs were first computed by discretizing the set of viewing directions. Recent algorithms take advantage of the fact that, except at critical viewing directions, the occluding contours of an object remain qualitatively (i.e., topologically) the same for small changes in the viewpoint (e.g., see [25]). In [17] a criticality-driven approach makes it possible to plan a sensorless sequence of squeezing operations to achieve some specified orientation of a polygonal part independent of its initial orientation.

3. VIRTUAL MANIPULATION SYSTEMS

Assume that we are given the description of a robot system and the design of an assembly product. We wish to plan an algorithm to make the robot system assemble the product from its individual parts. This problem can be formulated as a manipulation problem whose solution is a path in a large-dimensional space, the composite configuration space of the robot and all the movable parts [27]. A point (configuration) in this space fully represents a spatial placement of the robot and the other parts. A solution path describes all the motions that are necessary to construct the assembly. It is a continuous curve connecting an initial configuration where the parts are separated, to a goal configuration where they are assembled together according to the product design. The curve must satisfy certain physical constraints (e.g., the parts cannot move on their own). This view, however, makes assembly planning a highly complex motion planning problem. It also raises several conceptual issues that are still poorly understood, for example, the interaction between stability, fixturing, and grasping. Moreover, the manipulation system may not be known in advance. In fact, assembly planning is often used to help specify this system.

This leads to approaching assembly planning hierarchically, by solving a succession of simplified, but increasingly more realistic planning problems. The assembly algorithms for one problem are used to prune large subsets of the solution spaces of subsequent problems and to guide the search of these spaces. Each problem can then be seen as a planning problem for an abstraction of the (possibly not yet known) real manipulation system. We call this abstraction a *virtual manipulation system.* For example, at a high level of abstraction the parts composing the assembly are considered free-flying geometric objects; the corresponding constraints on assembly algorithms arise only from the product itself. At lower levels of abstraction, grippers, fixtures, and machines are introduced, along with uncertainties. As the virtual manufacturing system becomes more realistic, more detailed assembly algorithms can be generated. Just as one can write computer algorithms without knowing the details of their implementation, one can plan assembly algorithms without knowing which machines (or humans) will perform the manipulation.

Throughout the rest of this paper we will assume a very abstract virtual manufacturing system, in which individual parts are free-flying rigid geometric objects. This assumption corresponds to the most frequent situation in the early phase of the geometric design of a product and also underlies previous work in assembly sequencing (see Section 2). At this level of abstraction, we define an *assembly instruction* as the specification of the relative motions of m subsets of assembly parts, with $m \in \{2, 3, \ldots\}$, between an initial and a goal geometric arrangement of the parts. The m subsets are called the *moved sets* of the instruction. All the parts in the same moved set remain in constant relative position during the motion specified by the

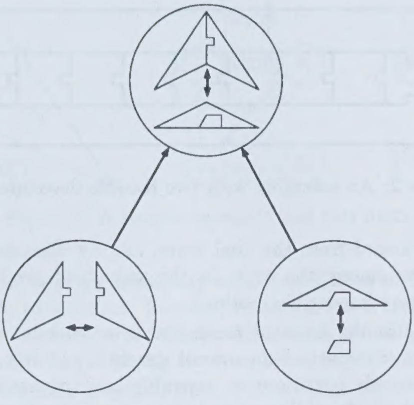


Figure 1: Tree of an assembly algorithm for a simple product

instruction; usually, each moved set forms a connected composite object, but this is not required.

An algorithm to assemble a product from its parts is a partial ordering of assembly instructions in time. The graph of this relation is a tree whose maximal element (the root) is the instruction generating the product itself (see Fig. 1). Furthermore, no two instructions that are not comparable in the ordering move the same assembly component. In the virtual assembly system considered here, an algorithm to disassemble a product is obtained by simply inverting the ordering of an algorithm assembling the product. An algorithm to service a product usually combines instructions disassembling and re-assembling subsets of a product. For simplification, but without loss of generality, in the rest of this paper we will only consider algorithms to assemble products, with initial arrangements where all the parts are separated. Such an algorithm is *correct* if all its instructions specify collision-free motions of the parts (contacts between parts are allowed, however) and the final relative positions of the parts are those specified in the geometric description of the product. An assembly algorithm of this form is often called an *assembly sequence.*

4. BLOCKING STRUCTURE OF AN ASSEMBLY

In assembly planning the goal state (the assembled product) is considerably more constrained than the initial state (the separated parts). This suggests that planning

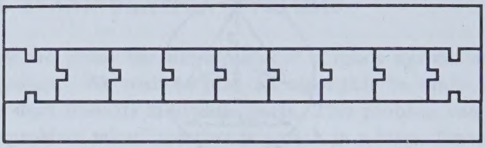


Figure 2: An assembly with two feasible decompositions

should proceed backward from the goal state, i.e., by disassembling the product. Indeed, the contacts among the parts in the assembled product can be used to quickly filter out many impossible motions.

Contact analysis is the basis of much previous work in assembly sequencing (e.g., [20]), where it was part of an overall generate-and-test planning approach. This approach recursively partitions an assembly into subassemblies and uses contact analysis to check the feasibility of each decomposition. However, the number of candidate decompositions is exponential in the number of parts, even though there may exist very few feasible ones. For example, in the assembly of Fig. 2, which consists of $n-2$ interlocking parts sandwiched between two plates, only two decompositions are feasible. The inherent inefficiency of generate-and-test led many authors to restrict their planners to assembly sequences in which a product is built by adding a single part at a time. With this restriction, generate-and-test has a lower-order exponential dependence on the number of parts. But it is often preferable to build subassemblies that are later merged into larger ones.

The inefficiency of generate-and-test derives from the fact that essentially the same contact analysis is done many times. This repetition can be avoided by performing a complete contact analysis first. The results, recorded in a compact data structure, can then be used to directly determine which assembly decompositions are feasible. This approach yields the concept of an NDBG. For clarity, we introduce a first simple type of NDBG below (2D with infinitesimal translations). More sophisticated NDBGs will be presented in Section 5. In particular, we will show that NDBGs are not limited to contact analysis.

4.1. DIRECTIONAL BLOCKING GRAPH

Consider a planar assembly A made of n polygonal parts $P_1, \ldots, P_n$. The interiors of any two parts in A are disjoint. If the boundaries of two parts intersect, the two parts are said to be in contact.

Suppose that we wish to remove one part, say P_i, by translating it along a direction defined by the unit vector d. We say that a part P_j of A ($j \neq i$) *blocks* the

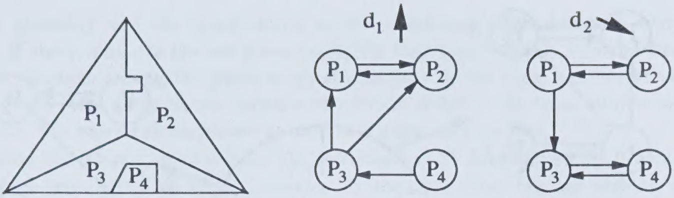


Figure 3: A simple assembly and two DBGs

translation of P_i along d if an arbitrarily small translation of P_i along d leads the interiors of P_i and P_j to intersect. Hence, if P_j blocks the translation of P_i, the two parts are necessarily in contact. A subassembly S of A is *locally free* to translate in direction d iff no part in $A \setminus S$ blocks the translation of any part of S along d.

The *directional blocking graph*, or DBG, $G(d, A)$ of A for an *infinitesimal translation* along d is a directed graph with nodes representing the parts of A. An arc connects P_i to P_j iff P_j blocks the translation of P_i along d. Fig. 3 shows a simple assembly made of 4 polygonal parts and its DBGs for infinitesimal translations along d_1 and d_2.

A subassembly S of A is locally free to translate in direction d iff no arcs in $G(d, A)$ connect parts in S to parts in $A \setminus S$. If $G(d, A)$ is strongly connected,[1] no such subassembly exists. Otherwise, at least one strong component of $G(d, A)$ must have no outgoing arcs. For example, in Fig. 3 the subassemblies $\{P_1, P_2\}$ and $\{P_1, P_2, P_3\}$ are locally free to translate in direction d_1; only the subassembly $\{P_3, P_4\}$ is locally free in direction d_2.

For a subassembly S to be assemblable with $A \setminus S$ by a translation along $-d$, it must be locally free to translate in direction d. This condition is necessary but not sufficient, since global accessibility is not considered. Also, merging S and $A \setminus S$ may require rotation. For those reasons, in Section 5 we will extend the above notions to non-infinitesimal translations and motions with rotation.

4.2. NON-DIRECTIONAL BLOCKING GRAPH

We represent the set of all translation directions by the unit circle $\mathcal{S}^1$. This circle is the locus of the extremity of the vector d when its origin is fixed.

Let P_i and P_j be two parts in contact. The set of directions in which P_i is locally

---

[1] A *strongly connected component* (or *strong component*) of a directed graph is a maximal subset of nodes such that for any pair of nodes (X_1, X_2) in this subset, a path connects X_1 to X_2. A graph is strongly connected if it has only one strong component.

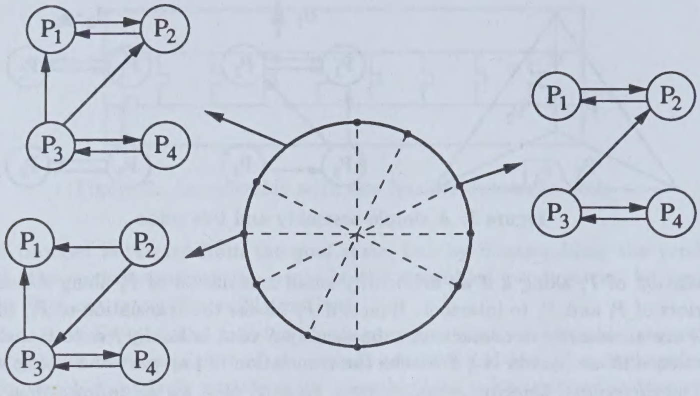


Figure 4: Part of the NDBG of the assembly of Fig. 3

free to translate relative to P_j is a closed cone (possibly a half-space or a single ray) [20, 48]. For every pair of parts P_i and P_j in contact in A, we draw the diameters of $\mathcal{S}^1$ parallel to the two sides of the cone characterizing the local freedom of P_i relative to P_j. The drawn diameters partition $\mathcal{S}^1$ into an arrangement of regions, the endpoints of the diameters and the open circular arcs between them. Every such region is *regular* in the sense that the DBG $G(d, A)$ remains constant when d varies over it. We denote the DBG of A for any direction in a regular region R by $G(R, A)$.

The arrangement of points and intervals on $\mathcal{S}^1$ and the associated DBGs form the *non-directional blocking graph* $\Gamma(A)$ of A for infinitesimal translations. It represents the blocking structure of A for infinitesimal translations in all directions.

Part of the NDBG of the assembly of Fig. 3 is shown in Fig. 4. The partition of $\mathcal{S}^1$ in this example consists of 20 regular regions.

4.3. COMPUTATION

Let us assume for simplification that the contacts between any two parts in A are edge-edge contacts (hence, there are no isolated contact points). Assume also that if a concave vertex of a part coincides with a convex vertex of another part, then the edges abutting these vertices are in pairwise contact. These assumptions eliminate cases requiring specific attention; however, these cases can be easily treated within the complexity bounds given below.

Let the input to the computation consist of the geometric model of the parts in the assembly and the specification of the contacting edges between every two parts. If these contacts are not given explicitly they can be easily inferred from the spatial relations among the parts in quadratic time in the total number of vertices of the parts [48]. Let n be the number of parts in A and c the total number of edge contacts. We represent each DBG as an $n \times n$ adjacency matrix.

Under the above assumptions, the partition of $\mathcal{S}^1$ is obtained by drawing the diameters parallel to all edge contacts. It contains $O(c)$ regular regions and is computed in $O(c \log c)$ time.

For each regular region R, we select an arbitrary direction d in R and compute the DBG $G(d, A)$. After clearing the adjacency matrix for the DBG $G(d, A)$, each edge contact is considered separately. For every edge contact between parts P_i and P_j, the inner product of d with the outer normal to P_i in the contacting edge with P_j is computed. If the inner product is strictly positive, an arc from P_i to P_j is added to $G(d, A)$ (if it does not already exist). The computation of $G(d, A)$ takes $O(n^2 + c)$ time. We can avoid the initial clearing of the adjacency matrix by keeping track of the updated cells in a stack of pointers (see [2], page 71). This modification reduces the computing complexity of $G(d, A)$ to $O(c)$. By repeating the computation for all regular regions, the NDBG $\Gamma(A)$ is constructed in $O(c^2)$ time. The size of $\Gamma(A)$ is $O(cn^2)$.

The above computation can be refined as follows. For any pair of parts in contact, if there are more than two edge contacts, we only retain the two diameters of $\mathcal{S}^1$ which bound the cone of directions in which one part is free to translate relative to the other. This yields a total of $O(r)$ diameters, where r is the number of pairs of parts in contact in A. The arrangement of $\mathcal{S}^1$ is computed in $O(c + r \log r)$ time. The resulting NDBG is computed in $O(c + r^2)$ time and has $O(rn^2)$ size. We always have $r \leq c$ and for many assemblies $r \ll c$. While c depends on both the number and the complexity of the parts of A, r only depends on their number. Although in the worst case $r \in O(n^2)$, it is often smaller.

The NDBG can also be incrementally updated as the product is being designed. Each modification can be treated as a combination of part deletions and part additions. The deletion of a part P is obtained by removing the node corresponding to P and all its adjacent arcs from every DBG, and merging regular regions that were previously separated by contacts involving P. It can be done in $O(r)$ time. Adding a part contacting r' parts already in the assembly can be done in $O(r'r)$ time.

4.4. GENERATING CANDIDATE ALGORITHMS

The above NDBG $\Gamma(A)$ is an implicit representation of a set of assembly algorithms in which every instruction merges exactly two subassemblies without rotation. This set contains all correct assembly algorithms; it may also contain incorrect algorithms, since global accessibility constraints were not considered to compute the NDBG. Hence the assembly algorithms given by $\Gamma(A)$ must be validated by further tests (not described here; see [48]). These tests operate on assembly operations; therefore each instruction in an assembly algorithm should be validated before constructing the rest of the algorithm. Since $\Gamma(A)$ allows the direct generation of instructions satisfying local freedom constraints, the number of additional tests required is usually relatively small.

Let a candidate algorithm be any algorithm contained in $\Gamma(A)$, and a candidate partitioning of A be two subassemblies, S and $A \setminus S$, such that one is locally free to translate in some direction d. Each DBG contains n nodes and at most $2r$ arcs. Hence, finding the strong components of any DBG takes $O(r)$ time [3] and generating a candidate partitioning of A, given $\Gamma(A)$, takes $O(r^2)$ time. Furthermore, if a candidate algorithm exists, one candidate algorithm also exists for each of the two subassemblies in any candidate partitioning. Thus, generating a candidate algorithm takes $O(r^2 n)$ time. Let u be the total number of candidate partitionings of A; $u \in O(2^n)$, but for most assemblies it is much smaller. The set of all candidate partitionings of A for some direction d is computed in $O(ru)$ output-sensitive time by reducing $G(d, A)$ to the acyclic graph of its strong components (see [48]). The set of all candidate partitionings of A is computed in $O(r^2 u)$ time.

4.5. COMPUTING VARIANT

One can notice that there is little or no change between the DBGs of two adjacent regular regions. Thus, once we have computed the DBG for one region, call it R_1, we can incrementally modify this graph to get the DBG for the next region in the NDBG list, instead of computing this DBG from scratch. We can proceed in this same way until all the regions have been considered.

To do this, we slightly modify the DBG of a region by attaching a *weight* to each arc of the graph. In $G(R_1, A)$, this weight is the number of inner products that were strictly positive in the above computation. The absence of an arc from P_i to P_j is treated as an arc of weight 0, and conversely. Let R_1 be a circular arc. The next region R_2 in the NDBG is necessarily a singleton. Let D be the diameter of $\mathcal{S}^1$ that ends at R_2, and $\{E_1, \ldots, E_s\}$, $s \geq 1$, be the set of all contact edges in A parallel to D. $G(R_2, A)$ can be derived from $G(R_1, A)$ by applying the following *crossing rule*:

> Initialize G to $G(R_1, A)$. For every contact edge E_k ($k = 1$ to s), let P_i and P_j be the two parts sharing this edge. If the inner product of any direction in R_1 and the outgoing normal to P_i in E_k is strictly positive, then retract 1 from the weight of the arc connecting P_i to P_j in G.

The graph G obtained at the end of the loop is $G(R_2, A)$. (Again, every arc weighted 0 is interpreted as no arc.)

If R_1 is a singleton and R_2 a circular arc, the crossing rule is similar:

> Initialize G to $G(R_1, A)$. For every contact edge E_k ($k = 1$ to s), let P_i and P_j be the two parts sharing this edge. If the inner product of any direction in R_2 and the outgoing normal to P_i in E_k is strictly positive, then add 1 to the weight of the arc connecting P_i to P_j in G.

Using these crossing rules and representing only one DBG at any one time allow us to successively compute all the DBGs in $O(r)$ amortized time after the $O(c + r \log r)$ computation of the partition of $\mathcal{S}^1$. Indeed, the cost of computing a DBG by applying the crossing rule is proportional to the number s of contact edges involved in the computation. This number is in $O(r)$, but throughout the computation of the entire NDBG, each edge is considered only twice. Hence, the time complexity of the construction of *all* the remaining DBGs is only $O(r)$.

A candidate partitioning can be computed in $O(r^2)$ time, and a candidate algorithm in $O(r^2 n)$ time by constructing the DBG of every generated subassembly. Compared to the algorithm of the previous two subsections, this computing variant allows substantial space saving, since it does not require representing all $O(r)$ DBGs at any given time. Moreover, if a computation (for example, evaluating a complexity measure) uses the DBGs, one at a time, in the same sequence as they are generated, this incremental technique also has greater time efficiency.

5. OTHER BLOCKING GRAPHS

The notion of an NDBG introduced in the previous section admits several extensions and variants. We present some of them below. See [48] for more detail.

5.1. 3D ASSEMBLIES

The NDBG for infinitesimal translations can be easily extended to 3D assemblies made of polyhedral parts. As in the 2D case, we simplify our analysis and eliminate cases requiring specific, but straightforward attention, by making the following assumptions: (1) the contacts between any two parts in A are face-face contacts; (2) if a concave edge of a part coincides with a convex edge of another part, then the faces bounded by these edges are in pairwise contact; (3) if a non-convex vertex of a part coincides with a vertex of another part, then the edges abutting these vertices are in pairwise contact. Let c be the total number of face-face contacts between parts of A.

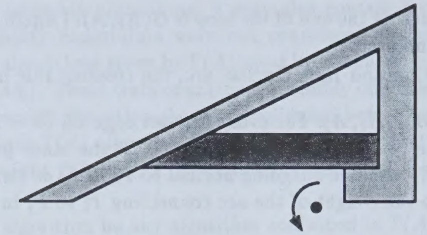


Figure 5: Need for rotation

The set of translation directions in 3D is represented by the unit sphere $\mathcal{S}^2$. The c plane-plane contacts induce arcs of great circles partitioning $\mathcal{S}^2$ into an arrangement of $O(c^2)$ regular regions of dimensions 2, 1 and 0 (faces, edges, and vertices, respectively). This arrangement is computed in $O(c^2)$ time using a topological sweep [12]. The NDBG is computed in $O(c^3)$ time and has size $O(c^2n^2)$. If only one DBG is represented at any one time, we can use crossing rules between regions and successively compute all the DBGs in $O(c^2)$ time. Generating a candidate partitioning of A into two subassemblies, given the NDBG, takes $O(rc^2)$ time, where r is the number of pairs of parts in contact. Generating a candidate assembly algorithm takes $O(rc^2n)$ time.

As in the 2D case, slightly lower complexity bounds can be obtained by considering each of the r pairs of parts in contact in sequence. For each pair, the set of directions in which one part is locally free to translate relative to the other is a convex cone. The intersection of this cone with the sphere $\mathcal{S}^2$ is a "polygon" bounded by arcs of great circles. Only the arrangement of regions created by these arcs need be used to compute the NDBG.

5.2. INFINITESIMAL GENERALIZED MOTIONS

One important extension is to allow motions in rotation. Fig. 5 shows a simple case where a part blocks another part for any infinitesimal translation, while a rotation is feasible.

Let us consider the 3D case only (the 2D case is just simpler), with polyhedral parts. The direction of an infinitesimal generalized motion (combining translation and rotation) is given by a unit vector in 6D. Hence, the set of all possible directions of motion make up the unit 5D sphere $\mathcal{S}^5$.

A Cartesian frame is attached to each part in A. For any two parts P_i and P_j,

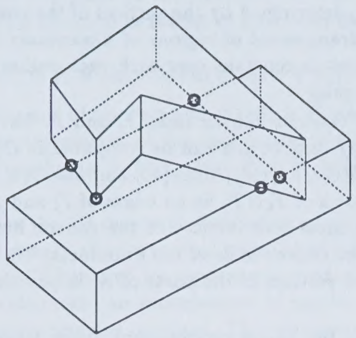


Figure 6: Plane-plane contact expressed as point-plane contacts

the configuration (position and orientation) of P_i relative to P_j is defined as the position and orientation of the frame attached to P_i relative to the frame attached to P_j. An infinitesimal generalized motion of P_i with respect to P_j is described by a 6D vector $dX = (dx, dy, dz, d\alpha, d\beta, d\gamma)$. The components dx, dy, and dz are those of the translation of the origin of the frame of P_i along the axes of the frame of P_j. The components $d\alpha$, $d\beta$, and $d\gamma$ are the infinitesimal angles by which the frame of P_i rotates about the axes of the frame of P_j.

Let V be a vertex of P_i. The motion described by dX causes V to undergo a translation $J_V\ dX$, where J_V is the 3×6 Jacobian matrix of the transform that gives the coordinates of V in the frame of P_j as a function of the configuration of P_i relative to P_j. Assume that P_i and P_j are in contact such that the vertex V of P_i is contained in the face F of P_j. Let n_F be the outgoing normal vector to F. The motion dX causes V to penetrate F when $n_F J_V dX < 0$, to break the contact with F when $n_F J_V dX > 0$, and to slide in F when $n_F J_V dX = 0$. The set of motions dX *allowed* by the contact between V and F are those satisfying $n_F J_V dX \geq 0$.

Now let P_i and P_j be two parts in contact such that a face F_i of P_i and a face F_j of P_j intersect at a piece of planar surface (plane-plane contact). The set of motions dX allowed by this contact is the intersection of all the closed half-spaces $n_{F_j} J_{V_k} dX \geq 0$ computed for the vertices V_k of the convex hull of the intersection of F_i and F_j [18, 49]. For example, in Fig. 6, the vertices V_k are circled.

We make the same simplifying assumptions about contacts as in Subsection 5.1. For each vertex V_k of the convex hull of the intersection of two parts, the equation $n_F J_{V_k} dX = 0$ defines a 5D hyperplane in the 6D space of infinitesimal generalized motions, which partitions $\mathcal{S}^5$ into two open half-spheres and a great circle. The set of all such hyperplanes, determined by the vertices of the convex hulls of the planar contacts, induces an arrangement of regions of dimensions $5, \ldots, 1, 0$ on $\mathcal{S}^5$. It is plain to see that the DBG is constant over each such region; hence, the regions of the arrangement are regular.

The intersection of two non-convex faces F_i and F_j having v_i and v_j vertices, respectively, has $O(v_i v_j)$ vertices and can be computed in $O(v_i v_j)$ time. Its convex hull is constructed in $\Theta(v_i v_j \log v_i v_j)$ time [40] and has $O(v_i + v_j)$ vertices. Indeed, all the intersection vertices of $F_i \cap F_j$ lie on edges of F_i and F_j, but each particular edge can contribute at most two vertices of the convex hull. Let k be the total number of vertices in the convex hulls of the c contacts. We have $k \in O(cv)$, where v is the total number of vertices in the parts of A; in practice, however, k is much smaller.

Since a singleton in the above arrangement arises whenever five hyperplanes intersect, the arrangement contains $O(k^5)$ regions. It is constructed in $O(k^5)$ time by a multi-dimensional topological sweep [12]. Constructing a DBG in any region is done in time $O(k)$. A crossing rule similar to the pure translational case can be established, yielding an $O(rk^5)$ time algorithm to build the NDBG for a 3D assembly, where $r \in O(n^2)$ is the number of pairs of parts in contact.

A necessary condition for a subassembly S to be assemblable with $A \setminus S$ is that there exists a DBG G such that no arcs in G connect parts in S to parts in $A \setminus S$. However, this condition is not sufficient. The above NDBG is thus an implicit representation of a set of assembly algorithms in which every instruction merges exactly two subassemblies. This set contains all correct assembly algorithms; it may also contain incorrect algorithms due to the fact that global accessibility constraints have not been considered.

Given the above NDBG, computing a candidate partitioning of A into two subassemblies takes $O(rk^5)$ time. Generating a candidate assembly algorithm takes $O(rk^5 n)$ time.

5.3. INFINITE TRANSLATIONS

To address global accessibility, we now present a variant of the NDBG that derives from the analysis of the interferences among parts for a family of non-infinitesimal motions: infinite translations. Given any two parts P_i and P_j in A (these two parts might not be in contact), we say that P_j blocks the infinite translation of P_i along d if the volume that P_i sweeps out when it translates along d from its initial position in A to infinity intersects P_j. The notions of a DBG and an NDBG for infinite translations follow from this blocking relation.

Let P_i and P_j be any two parts in the assembly A. The set $\mathcal{B}_{ij}$ of directions d along which P_j blocks P_i is identical to the set of directions along which the "grown" object

$$P_j \ominus P_i = \{a_j - b_i \mid a_j \in P_j, b_i \in P_i\},$$

i.e., the Minkowski difference of the two sets of points P_j and P_i at their positions in A, blocks the infinite translation of the coordinate origin O. If P_i and P_j are polygons in 2D (resp. polyhedra in 3D), then $P_j \ominus P_i$ is also a polygon (resp. a polyhedron) [27, 32].

Again, let us consider the 3D case with polyhedral parts. $\mathcal{B}_{ij}$ is the intersection of $\mathcal{S}^2$ and the polygonal cone of all rays erected from O and intersecting $P_j \ominus P_i$. This intersection is a region of $\mathcal{S}^2$ bounded by segments of great circles. The great circles supporting these segments create an arrangement of regular regions overwhich the DBG for infinite translations remains constant. This arrangement and the associated DBGs form the NDBG of the assembly for infinite translations.

A sufficient condition for a subassembly S to be directly assemblable with $A \setminus S$ is that there exists a DBG G in the above NDBG such that no arcs in G connect parts in S to parts in $A \setminus S$. This condition is not necessary since, for instance, a path with multiple extended translations may be required. The above NDBG is an implicit representation of all correct assembly algorithms in which every instruction merges two subassemblies by a single translation.

Let v be the total number of vertices in the parts of A. The arrangement on $\mathcal{S}^2$ has size $O(v^4)$ and can be computed in $O(v^4)$ time. The NDBG is computed in $O(n^2v^4)$ time. Each DBG has $O(n^2)$ arcs, so that finding its strong components takes $O(n^2)$ time. Hence, deciding whether A can be broken down into two subassemblies that can be merged by a single translation (and constructing one such partitioning, if one exists) takes $O(n^2v^4)$ time. Generating an assembly algorithm takes $O(n^3v^4)$ time.

5.4. DISCUSSION

One can easily imagine other NDBGs. For example, one may consider the case where an assembly is constructed by merging more than two, say $m + 1$, subassemblies at a time. The corresponding NDBG for infinitesimal translations in 3D would be obtained by partitioning the $(3m - 1)$-dimensional sphere $\mathcal{S}^{3m-1}$, representing m simultaneous translation directions, into an arrangement of regular regions.

NDBGs for infinite generalized motions and, more generally, sequences of extended motions along different directions could be investigated as well. The investigation for sequences of extended translations is under way in [33]. It leads to partitioning the composite configuration space $\mathcal{C}_A$ of the parts in A into regions over which all the relative placements of the parts induce the same NDBGs for infinitesimal translations. This construct yields a collection of NDBGs distributed over regions of $\mathcal{C}_A$. A careful representation of $\mathcal{C}_A$ yields a polynomial collection of NDBGs. From there, the generation of a partitioning of A for a bounded-length sequence of extended translations takes polynomial time. The construct can be generalized to sequences of extended generalized motions.

Each NDBG is defined for a limited family of motions and describes only the class of assembly algorithms for these motions. NDBGs for infinitesimal motions are underconstrained and may contain incorrect algorithms. NDBGs for infinite motions or bounded-length sequences of extended motions are overconstrained and may not include all possible algorithms. Trying to construct and exploit a unique NDBG or collection of NDBGs covering all possible motions would bring us back to the general planning problem of finding a coordinated path for a set of parts. This problem is known to be PSPACE-hard [23, 37] and is strongly believed to require exponential time in the number of parts.

Polynomial NDBGs such as those described above should be regarded as efficient filters to quickly identify feasible assembly operations and algorithms (for instance, using the NDBG for infinite translations) and eliminate infeasible ones (with the NDBG for infinitesimal rigid motions). When required, the candidate algorithms generated by the latter can be analyzed further using more general, but also more expensive, motion planning techniques.

Allowing more than two subassemblies to be merged simultaneoulsy and/or accepting sequences of extended motions yield NDBGs that are considerably more expensive to compute. An interesting development would therefore be to first construct the simplest NDBGs and then extend them locally (that is, for the subsets of parts that require the extension) when they do not allow the generation of assembly algorithms.

6. IMPLEMENTATION

We have implemented the algorithms constructing the NDBGs for infinitesimal and infinite translations both in 2D and 3D [48]. In 3D our implementation allows parts with planar, cylindrical, and some helicoidal faces. For infinitesimal generalized motions, we have implemented a hybrid algorithm that has the same time complexity as the translational version. This algorithm considers all pure translations, plus some "suggested generalized motions" inferred from nonplanar contacts in the assembly. For example, a cylindrical contact suggests pure rotation about its axis; a threaded contact between two helicoidal surfaces suggests a screwing motion; etc. The set of suggested generalized motions is incomplete but accounts for most motions required

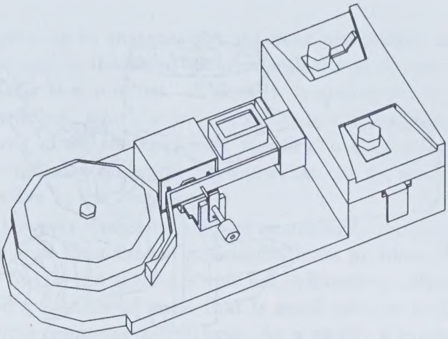


Figure 7: Electric bell

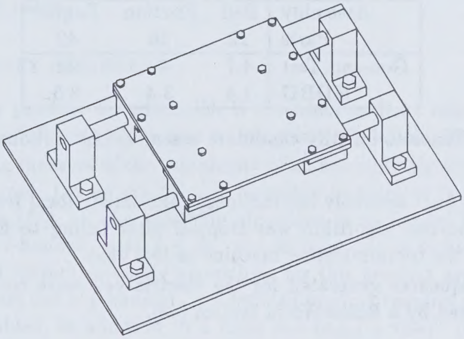


Figure 8: Friction-testing machine

by actual assembly products.

Our programs are written in CommonLisp as part of a larger assembly sequencing system [48] and run on a DEC 5000 workstation. We have used the NDBGs to compute candidate partitionings of assemblies and construct candidate assembly algorithms. We ran the programs on a variety of assemblies including a 22-part electric bell (Fig. 7), a friction-testing machine with 36 parts (Fig. 8), and the 42-part model-aircraft combustion engine (Fig. 9). Table 1 gives the running times of both the generate-and-test approach and the NDBG-based methods to identify all parti-

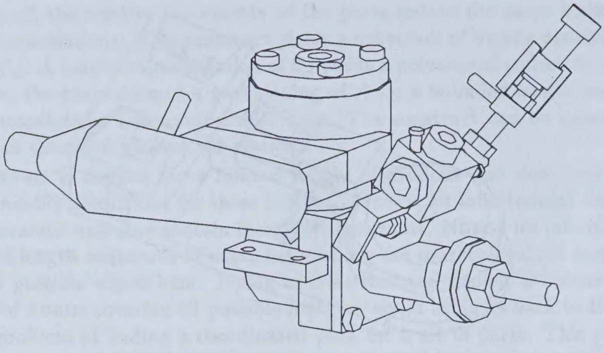


Figure 9: Model-aircraft engine

Assembly Parts	Bell 22	Friction 36	Engine 42
Gen-and-Test	4.7	—	74
NDBG	1.4	3.4	8.5

Table 1: Times to identify candidate assembly instructions, in seconds

tionings (hence, final assembly instructions) that satisfy local freedom constraints. The generate-and-test algorithm was stopped after failing to find any candidate instructions for the friction-testing machine in two days.

Assembly sequences generated for the electric bell were converted into robot programs executed by a RobotWorld system [43].

7. COMPLEXITY OF AN ASSEMBLY

The outcome of assembly planning, i.e., assembly algorithms, can be used to specify, select, design, and/or program the manipulation systems that will execute these algorithms. Another goal which currently attracts increasing interest is to use this outcome to help designers create products that are easier to manufacture and service. The problem, however, is that there may be many possible assembly algorithms, especially when the manipulation system is loosely defined. Showing the most promising algorithms (for some criterion), e.g., by displaying a graphic simulation of their execution, would certainly be useful. But it would have to be limited to a few milestones of the design process in order to avoid taking too much of the designers' time.

Instead, we propose to measure the inherent complexity of a product (at its current stage of design) as the lower-bound complexity of all the assembly algorithms that are feasible for this product. This idea is analogous to the computational complexity of a problem, which is defined as the lower-bound complexity (in terms of time and space) of all the computer algorithms that solve the problem [16]. The estimated complexity measures for a product can be used to provide concise, pertinent information to the designers.

To be useful, however, the complexity of an assembly product must be measured along many more axes than that of a computational problem. We propose several such axes below, but by no means is this list exhaustive. Moreover, assembly algorithms apply to a number of parts that is small relative to the amount of data processed by typical computer algorithms. As a result, a much tighter evaluation of assembly complexity is required than the classical asymptotic analysis applied to computer algorithms. We will show how the NDBGs make it possible to perform this analysis in many cases.

7.1. COMPLEXITY MEASURES

Let an assembly product be *admissible* if it admits at least one correct assembly algorithm. In the following discussion we consider only admissible assemblies.

An interesting measure of the complexity of an assembly algorithm is the number of hands it requires. Let us say that an assembly instruction is ℓ-handed if it has $\ell + 1$ moved sets ($\ell \geq 1$). An assembly algorithm is m-handed if all its assembly instructions are ℓ-handed, with $\ell \leq m$, and at least one is m-handed. A product is *p-handed* if all correct assembly algorithms for this product are m-handed, with $m \geq p$, and at least one is p-handed. A p-handed product requires p moving "hands" to be (dis)assembled, in addition to a fixed one (e.g., a vise). It is shown in [37] that an assembly made of n parts may require up to $n - 1$ moving hands to be assembled (i.e., the assembly can only be built by moving every part relative to all the other parts simultaneously). However, it is in general much more cost-effective to manufacture and service a 1-handed product than a multi-handed one. The class of 1-handed products is thus an important one.

Let a *subassembly* be any subset of parts in their final relative positions. An assembly algorithm is *monotonic* if each of its instructions merges the corresponding moved sets into a subassembly [48, 51]. A product is *p-handed monotonic* if it admits a p-handed monotonic assembly algorithm. For example, the 3-part latch assembly shown in Fig. 10 is not 1-handed monotonic, but is 2-handed monotonic. It is clear that any admissible n-part assembly is $(n - 1)$-handed monotonic; however,

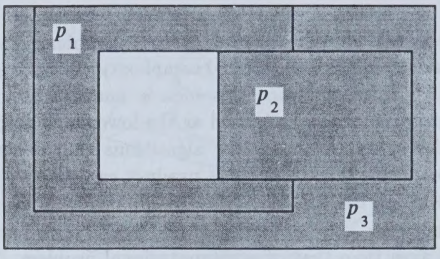


Figure 10: Three-part latch assembly

there exist p-handed n-part products, with $p < n-1$, that are not $(p+i)$-handed monotonic, for any $i \in [0, n-p-2]$. A non-monotonic assembly (for some given number of hands) requires bringing parts into intermediate relative positions (where they are ungrasped) that have to be changed later (which requires regrasping parts). It may require many more motions than a monotonic assembly with the same number of parts. So, the class of 1-handed monotonic products is an important subclass of the 1-handed products.

The nature and the number of degrees of freedom required to perform the motions specified by the assembly instructions are another key measure of product complexity. Let an assembly instruction be *m-prismatic* if the motion of every moved set is a sequence of at most m extended translations. Let a product be p-prismatic if it admits an assembly algorithm in which all instructions are m-prismatic, with $m \leq p$, for some initial arrangement of its parts in which every two parts are separated, and one instruction is p-prismatic. Since single translations are much more cost-effective to generate than multiple translations and generalized motions, the class of 1-prismatic products is another important one. One can further characterize the complexity of a product within this class by the minimal number of translation directions that are needed. A *stack* product is a 1-prismatic 1-handed monotonic product that admits an algorithm in which all instructions specify translations along the same direction. Large subassemblies of many small consumer electronic products are stack products. The concepts of prismatic and stack products can be easily extended to allow for screws in the products.

The length of the longest sequence of instructions in an assembly algorithm, the *length* of the algorithm, is another important attribute of this algorithm. Indeed, assuming that the time taken by any assembly instruction is lower and upper bounded regardless of the moved sets, it directly impacts the total execution time. The length of the shortest correct algorithm for a product is therefore a pertinent measure of the complexity of this product. It characterizes the extent to which the product can

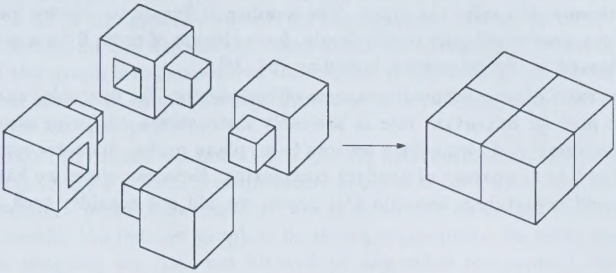


Figure 11: Nonlinearizable 1-handed assembly

be broken down into subassemblies that can be manufactured concurrently.

However, some mass-produced assemblies are more cost-effective to manufacture by bringing one part at a time. This concept can be captured as follows. Let us say that an instruction is *linear* if it is 1-handed and one of its moved sets is a single part. A linear algorithm consists of linear instructions only, hence is a total ordering. A product that admits a linear algorithm is said to be *linearizable.* Many 1-handed products are not linearizable (see one example in Fig. 11) and are in general less amenable to mass-production assembly lines. A refinement of this concept is to measure the minimal number p of nonlinear instructions over all possible 1-handed assembly algorithms; the minimal number of separate assembly lines needed for the product is $p+1$.

Moved sets need to be grasped or fixtured in a way that parts in the same set cannot move relative to each other under a variety of forces (gravity, acceleration, contact). This is achieved by positioning fingers or fixture elements on their boundary. A grasp of a moved set achieves *form closure* if, when the fingers are locked relative to each other, no part in the moved set can move relative to any other and the fingers [26, 27]. (Another concept, involving friction and forces, is that of force closure; we will not discuss it here, though it is perhaps more practical than form closure.) Given some abstract dimensionless model of a finger, e.g., a point-plane contact, a measure of the complexity of an assembly instruction is the number of fingers required to achieve a form-closure grasp of all its moved sets. Weaker notions can also be considered. For instance, a grasp achieves *prismatic form-closure* if it prevents every part to move in translation relative to any other. A grasp achieves *form closure of order* q if it prevents up to $q+1$ subassemblies to move relative to each other. Thus, a form-closure grasp of order 0 assumes that no internal motion in the moved set is possible; a form-closure grasp of order 1 assumes that a single internal motion (along any direction) is possible; etc. The higher the degree of form-closure, the safer the grasp. The number of fingers to achieve form-closure grasps for a *single* rigid part (equivalently, form-closure of order 0 for a set of parts) is investigated in several papers, including [34, 36].

There exist other pertinent measures of complexity. For example, uncertainties may also play an important role in assembly instructions, requiring sensors to be used. The number of elementary sensors (e.g., plane probe, diameter sensors) may also be used as a measure of product complexity. However, since we have ignored physical and uncertainty issues in this paper, we will not consider such a measure further.

7.2. EVALUATION OF COMPLEXITY MEASURES

We now show how NDBGs can be used to estimate complexity measures of a product. We mainly focus on the NDBG for infinitesimal translations in 3D, but a similar development can be done with other NDBGs. We evaluate the time complexity for evaluating some complexity measures assuming the NDBG as the input to this computation. However, several measures can be computed while the NDBG is constructed, within the time complexity required by this construction; some can even be more efficiently obtained without computing the NDBGs.

Let us say that a 1-handed assembly algorithm is *correct for infinitesimal translations* if, for every instruction, one moved set is locally free to translate in some direction d in the final arrangement specified by the instruction. The set of assembly algorithms that can be extracted from the NDBG $\Gamma(A)$ of an assembly A for infinitesimal translations includes all correct 1-handed monotonic prismatic assembly algorithms, the importance of which was discussed above. If no assembly algorithm can be extracted from $\Gamma(A)$, it is guaranteed that the assembly is not 1-handed monotonic prismatic.

If a candidate algorithm can be extracted from $\Gamma(A)$, one may wish to know if the assembly is linearizable for translations. Checking the existence of a candidate partitioning of A into S and $A \setminus S$, where S is a singleton, and finding one such partitioning, if one exists, take the same time, $O(rc^2)$, as checking the existence of any candidate partitioning (see Subsection 5.1). In addition, if an assembly is linearizable, all its subassemblies are also linearizable. Hence, checking that a candidate linear algorithm exists takes $O(rc^2n)$ time. If none exists, the product is not linearizable for translations. (This check is a case where using the NDBG is not very pertinent. Indeed, a candidate linear algorithm can be directly computed from the description of the c contacts in only $O(nc \log c)$ time.)

The minimal length of all the candidate algorithms for a product is a lower bound on the minimal length of all correct one-handed monotonic prismatic assembly algorithms. The set of all candidate algorithms can be represented as an AND/OR graph [21] and searched for the shortest one using alpha-beta pruning [38]. But in general, the size of this graph is exponential in the number n of parts. Efficiently computing the minimal length of candidate algorithms, or a bounded approximation of this length, is still an open problem.

The NDBG $\Gamma(S)$ of any subassembly $S \subseteq A$ for infinitesimal translations can also be used to generate a prismatic form-closure grasp of order 1 for S. Let each finger be modeled by a point-plane contact. We scan all the DBGs in $\Gamma(S)$. For every DBG, we consider the reduced graph of its strong components; for every component C with no outgoing arc (i.e., not blocked by any other component), we place a dimensionless finger f in a face F of C whose external normal vector has positive inner product with d and we update $\Gamma(S)$ by adding f to the parts of S, that is, by adding the plane parallel to F to the partition of $\mathcal{S}^2$. At the end, the arrangement in $\mathcal{S}^2$ must be such that, in every DBG, each strongly connected component with no outgoing arc contains one finger. In general, the number of fingers in the generated grasp is not minimal to achieve prismatic form-closure of order 1; it nevertheless gives a good indication of the cost of grasping or fixturing S.

The set of assembly algorithms that one can extract from the NDBG of A for infinitesimal generalized motions includes all correct 1-handed monotonic assembly algorithms. Various complexity measures can be extracted as above.

The set of assembly algorithms that can be extracted from the NDBG of A for infinite translations is the set of all the 1-handed monotonic 1-prismatic assembly algorithms for A. This NDBG allows the computation of various complexity measures for this family of algorithms. In particular, A is a stack product exactly when an algorithm can be extracted from a single DBG. Such a DBG must be an acyclic graph (this is a necessary and sufficient condition), which can be checked in $O(n^2)$ time. Hence, determining whether A is a stack product takes $O(n^2v^4)$ time.

Notice that no combination of the previous NDBGs allows one to decide in all cases whether a product is admissible.

7.3. EXAMPLE

We have implemented several algorithms to evaluate complexity measures of a product from its NDBG for infinitesimal translations and suggested rotations (see Section 6). We applied these algorithms to a 23-part simplified version of the engine shown in Fig. 9. The results are as follows:

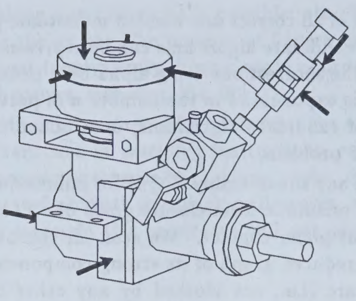


Figure 12: Form-closure grasp of order 1 of the engine

Number of hands and monotonicity. The NDBG allows the generation of candidate algorithms. Hence, the engine admits 1-handed monotonic algorithms that are correct for infinitesimal motions.

Is it prismatic? The engine contains a number of threaded contacts, so it is not prismatic. If these threaded contacts are replaced by cylindrical ones, the existence of candidate algorithms in the modified NDBG suggests that there may exist 1-handed monotonic prismatic algorithms.

Is it linearizable? As far as infinitesimal motions are concerned, the product is linearizable. This computation was performed with the faster method mentioned above, not directly using the NDBG.

Minimal length. The minimum-length algorithm contained in the NDBG has depth 6. This was computed by searching the AND/OR graph representing all 1-handed monotonic assembly algorithms for the engine. The graph has 1027 nodes (each distinct subassembly is represented by a single node) and required 14 minutes to build and search. Better algorithms are needed for products having many more parts.

Is it a stack product? The engine requires infinitesimal translation in several directions to assemble, so it cannot be a stack product.

Number of fingers. Fig. 12 shows a prismatic form-closure grasp of order 1 computed for the simplified engine. The grasp consists of 10 fingers. The contact points of 8 fingers are shown as arrows; two additional fingers are required to hold a loose part inside the engine. It may seem that an additional finger is needed to support the assembly from below. In fact, the finger pointing up and left at the top-right achieves that function.

The implemented algorithms illustrate the kind of information which one may extract from NDBGs. Some of them, however, are based on inefficient brute-force methods that are not practical for complex products. Furthermore, no algorithms have been implemented yet to exploit NDBGs for infinite translation or any other family of extended motions. It is not obvious that polynomial NDBGs will allow all interesting complexity measures to be computed in polynomial time. Very likely, for some measures, we will have to content ourselves with approximate algorithms. Incremental algorithms operating in marginal time while the product is being designed will also be useful.

8. CONCLUSION

This paper presented techniques to reason about mechanical assembly. We investigated the following two problems: (1) the automatic generation of assembly algorithms from CAD data, and (2) the characterization of the complexity of assembly designs. Our main goal was to provide efficient computational support to help designers create products that are easier to manufacture and service. The proposed techniques nevertheless have broader application. They could also be used, for example, to plan the actions of autonomous maintenance robots.

One contribution of this paper consists of the notion of an NDBG and the algorithms to construct it. The construction of an NDBG is based on an analysis of the interferences (blocking relations) among the parts in an assembly. Several variants were presented for different families of motion inducing different blocking relations. Unlike previous methods, the NDBG allows assembly sequences to be computed in time polynomial in the complexity of the parts of the assembly.

Another contribution of the paper is the notion of the algorithmic complexity of an assembly product, the set of complexity measures proposed, and the techniques to estimate them from precomputed NDBGs. Our goal here was to concisely assess the difficulty, hence the cost, of (dis)assembling a product.

Computational complexity has been instrumental to identify the strengths and weaknesses of particular computer algorithms, and classify problems. It has contributed enormously to the progress of computer technology. We think that algorithmic complexity measures for assembly products could have a similar impact on assembly products and processes, bringing improvements that would outpace any advances in the underlying manufacturing technologies themselves. We also expect that it will contribute to the emergence of a completely new type of manufacturing technology where a large number of very simple, mass-produced actuators and sensors are combined in highly modular manufacturing systems. The RISC approach to robotics recently proposed in [8] takes a similar view.

One of our current research goals is to construct polynomial NDBGs for more complex families of motion and investigate the effect of additional constraints (e.g., uncertainty, grasp accessibility) on the construction of NDBGs. Another goal is to investigate further the exploitation of NDBGs and design a more comprehensive set of efficient complexity evaluation methods. On the experimental side, we are currently integrating our algorithms into a highly interactive design-assistant system.

References

[1] — 1990. Special Issue on Robotic Assembly and Task Planning. *AI Magazine*, AAAI Press, 11(1).

[2] Aho, A.V., Hopcroft, J.E., and Ullman J.D. 1974. The Design and Analysis of Computer Algorithms. Reading, MA: Addison-Wesley.

[3] Aho, A.V., Hopcroft, J.E., and Ullman J.D. 1983. Data Structures and Algorithms. Reading, MA: Addison-Wesley.

[4] Arkin, E.M., Connelly, R., and Mitchell, J.S. 1989. On Monotone Paths among Obstacles with Applications to Planning Assemblies. *ACM Symp. on Computational Geometry*, 334–343.

[5] Baldwin, D.F. 1990. Algorithmic Methods and Software Tools for the Generation of Mechanical Assembly Sequences, Master's thesis, MIT.

[6] Boothroyd, G. 1991. Assembly Automation and Product Design. Marcel Dekker, Inc., New York, NY.

[7] Bourjault, A. 1984. Contribution à une Approche Méthodologique de l'Assemblage Automatisé: Elaboration Automatique des Séquences Opératoires. PhD thesis, Faculté des Sciences et des Techniques de l'Université de Franche-Comté, France.

[8] Canny, J.F. and Goldberg, K.Y. 1992. RISC Robotics. in preparation.

[9] Cutkosky, M.R. and Tenenbaum, J.M. 1990. A Methodology and Computational Framework for Concurrent Product and Process Design. *ASME J. of Mechanism and Machine Theory*, 25(3):365–381.

[10] De Fazio, T.L. and Whitney, D.E. 1987. Simplified Generation of All Mechanical Assembly Sequences. *IEEE J. of Robotics and Automation*, RA-3(6):640–658. Errata in RA-4(6):705-708.

[11] de Kleer, J. and Williams, B.C. (guest editors). 1991. Special Volume, Qualitative Reasoning about Physical Systems II. *Artificial Intelligence*, 51(1–3).

[12] Edelsbrunner, H. and Guibas, L. 1989. Topologically Sweeping an Arrangement. *J. of Computer and System Sciences*, 38:165–194.

[13] Fahlman, S.E. 1974. A Planning System for Robot Construction Tasks. *Artificial Intelligence*, 5(1):1–49.

[14] Faltings, B. 1992. A Symbolic Approach to Qualitative Kinematics. *Artificial Intelligence*, 56(2–3), 139–170.

[15] Fikes, R.E. and Nilsson, N.J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3-4):189–208.

[16] Garey, M.R. and Johnson, D.S. 1979. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. New York, NY:W.H. Freeman and Co.

[17] Goldberg, K.Y. 1990. **Stochastic Plans for Robotic Manipulation**. PhD Thesis, Dept. of Computer Science, Carnegie Mellon University.

[18] Hirukawa, H., Matsui, T., and Takase, K. 1991. A General Algorithm for Derivation and Analysis of Constraint for Motion of Polyhedra in Contact. *IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, Osaka, 38–43.

[19] Hoffman, R.L. 1991. A Common Sense Approach to Assembly Sequence Planning. In Homem de Mello, L.S. and Lee, S., editors, **Computer-Aided Mechanical Assembly Planning**, Boston, MA: Kluwer Academic Publishers, 289–314.

[20] Homem de Mello, L.S. 1989. **Task Sequence Planning for Robotic Assembly**. PhD Thesis, Carnegie Mellon University.

[21] Homem de Mello, L.S. and Sanderson, A.C. 1991. Representations of Mechanical Assembly Sequences. *IEEE Tr. on Robotics and Automation*, 7(2):211–227.

[22] Homem de Mello, L.S. and Lee, S. (editors) 1991. **Computer-Aided Mechanical Assembly Planning**. Boston, MA: Kluwer Academic Publishers.

[23] Hopcroft, J.E., Schwartz, J.T., and Sharir, M. 1984. On the Complexity of Motion Planning for Multiple Independent Objects: PSPACE-Hardness of the 'Warehouseman's Problem', *The Int. J. of Robotics Research*, 3(4):76–88.

[24] Joskowicz, L. and Sacks, E. 1991. Computational Kinematics. *Artificial Intelligence*, 51(1–3):381–416.

[25] Kriegman, D.J. and Ponce, J. 1990. Computing Exact Aspect Graphs of Curved Objects: Solids of Revolution. *Int. J. of Computer Vision*, 5(2):119–135.

[26] Lakshminarayana, K. 1978. The Mechanics of Form Closure. *ASME Paper*, No. 78-DET-32.

[27] Latombe, J.C. 1991. **Robot Motion Planning**. Boston, MA: Kluwer Academic Publishers.

[28] Lee, S. and Shin, Y.G. 1990. Assembly Planning Based on Geometric Reasoning, *Computation and Graphics*, 14(2):237–250.

[29] Liebermann, L.I. and Wesley, M.A. 1977. AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly. *IBM J. of Research and Development*, 21(4):321–333.

[30] Liu, Y. 1990. **Symmetry Groups in Robotic Assembly Planning**. PhD Thesis, COINS Tech. Rep. 90-83, Computer and Information Science Dept., University of Massachusetts, Amherst.

[31] Lozano-Pérez, T. 1976. **The Design of a Mechanical Assembly System**. Tech. Rep. AI-TR 397, AI Lab., MIT.

[32] Lozano-Pérez, T. 1983. Spatial Planning: A Configuration Space Approach, *IEEE Tr. on Computers*, 32(2):108–120.

[33] Lozano-Pérez, T. and Wilson, R.H. 1992. Assembly Sequencing. In preparation.

[34] Markenscoff, X., Ni, L., and Papadimitriou, C.H. 1990. The Geometry of Grasping. *The Int. J. of Robotics Research*, 9(1):61–74.

[35] McDermott, D.V. 1992. Robot Planning. *AI Magazine*, AAAI Press, 13(2):55–79.

[36] Mishra, B., Schwartz, J.T., and Sharir, M. 1987. On the Existence and Synthesis of Multifinger Positive Grips. *Algorithmica*, 2:541–558.

[37] Natarajan, B.K. 1988. On Planning Assemblies. *ACM Symp. on Computational Geometry*, 299–308.

[38] Nilsson, N.J. 1980. **Principles of Artificial Intelligence**, Los Altos, CA: Morgan Kaufmann.

[39] Pollack, R., Sharir, M., and Sifrony, S. 1988. Separating Two Simple Polygons by a Sequence of Translations, *Discrete and Computational Geometry*, 3:123–136.

[40] Preparata, F.P. and Shamos, M.I. 1985. **Computational Geometry: An Introduction**. New York, NY:Springer-Verlag.

[41] Reif, J.H. 1979. Complexity of the Mover's Problem and Generalizations. *20th IEEE Symp. on Foundations of Computer Science*, 421–427.

[42] Sacerdoti, E.D. 1977. **A Structure for Plans and Behavior**. Amsterdam: Elsevier.

[43] Scheinman, R. 1988. RobotWorld: A Multiple Robot Vision Guided Assembly System. In Bolles, R.C. and Roth, B., editors, **Robotics Research**, Cambridge, MA: MIT Press, 23–27.

[44] Subramani, A. 1992. **Development of a Design for Service Methodology**. PhD Thesis, Dept. of Industrial and Manufacturing Engineering, The University of Rhode Island, Kingston, RI.

[45] Taylor, R.H. 1976. **Synthesis of Manipulator Control Programs from Task-Level Specifications**. PhD Thesis, Dept. of Computer Science, Stanford University.

[46] Toussaint, G.T. 1985. Movable Separability of Sets. In Toussaint, G.T., editor, **Computational Geometry**. Amsterdam: Elsevier.

[47] Wilson, R.H. and Rit, J.F. 1990. Maintaining Geometric Dependencies in an Assembly Planner. *IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 890–895.

[48] Wilson, R.H. 1992. **On Geometric Assembly Planning**. PhD Thesis, Dept. of Computer Science, Stanford University.

[49] Wilson, R.H. and Matsui, T. 1992. Partitioning an Assembly for Infinitesimal Motions in Translation and Rotation. *IEEE Int. Conf. on Intelligent Robots and Systems*, Raleigh, NC, 1311–1318.

[50] Wilson, R.H. and Latombe, J.C. 1992. On the Qualitative Structure of a Mechanical Assembly. *AAAI-92 Nat. Conf. on Artificial Intelligence*, San Jose, CA, 697–702.

[51] Wolter, J.D. 1988. **On the Automatic Generation of Plans for Mechanical Assembly**. PhD thesis, The University of Michigan.

[52] Wolter, J., Chakrabarty, S., and Tsao, J. 1992. Mating Constraint Languages for Assembly Sequence Planning. *IEEE Int. Conf. on Robotics and Automation*, Nice, France, 2367–2374.

INFORMATION-THEORETIC CAPACITY OF RANDOM MULTI-RECEIVER NETWORKS

S.V. Hanly[1]

Statistical Laboratory, University of Cambridge, 16 Mill Lane, Cambridge CB2 1SB, UK. tel: +44 0223 337944. fax:+44 0223 337956.

P. Whiting

Department of Electronic and Electrical Engineering, Royal College, University of Strathclyde, 204 George St., Glasgow G1 1XW, Scotland. tel: +44 041 552 4400. fax: +44 041 552 2487.

ABSTRACT

We consider multi-receiver networks with diversity reception from an information-theoretic point of view. We show that the information-theoretic notion of capacity emerges as a law of large numbers in a sequence of random networks. We then use the theory to gain insights into how spectrum should be allocated in mobile radio.

INFORMATION THEORY; LAW OF LARGE NUMBERS; DIVERSITY; MOBILE RADIO; SPREAD SPECTRUM; NETWORK CAPACITY;

1 Introduction

In this paper we find the information-theoretic capacity region of a multi-receiver network. The motivation is provided by the uplink of a mobile radio communication network. As a first step to modelling fading effects and power control, we consider a sequence of random networks and show that a capacity emerges as a law of large numbers. We then consider insights that are obtainable from the information-theoretic analysis: we show that frequency reuse partitioning is suboptimal and that spread spectrum schemes can approach capacity. We also see that as the scale of a system increases, the proportion of capacity needed to be set aside to deal with interference fluctuations decreases. This economy of large scale is a feature of spread spectrum systems.

2 Some Results from Information Theory

The capacity of the continuous time bandwidth constrained Gaussian channel is well known see [8]

$$C = W \log_2 \left(1 + \frac{P}{\eta W}\right) \tag{1}$$

where $[-W, \cdots W]$ is the bandwidth, P is the received power and $\frac{\eta}{2}$ is the noise spectral density.

---

[1] Research funded by the Commonwealth Scholarship and fellowship plan

The extension to a network of K receivers is given in [6]. Let the received powers at the K receivers be $P_1, \cdots P_K$. Define the "power profile" matrix $\mathcal{P}$ by

$$\mathcal{P} = \begin{bmatrix} P_1 & \sqrt{P_1 P_2} & \cdots & \sqrt{P_1 P_K} \\ \sqrt{P_1 P_2} & P_2 & \cdots & \sqrt{P_2 P_K} \\ \vdots & \vdots & \vdots & \vdots \\ \sqrt{P_1 P_K} & \sqrt{P_2 P_K} & \cdots & P_K \end{bmatrix} \tag{2}$$

Then the capacity is ([6])

$$C = W \log_2 \left(|I + \frac{1}{\eta W} \mathcal{P}| \right) = W \log_2 \left(1 + \sum_{j=1}^{K} \frac{P_j}{\eta W} \right) \tag{3}$$

where | . | is the determinant operator. Here we are assuming K-receiver macrodiversity is employed. The situation is depicted in figure 1.

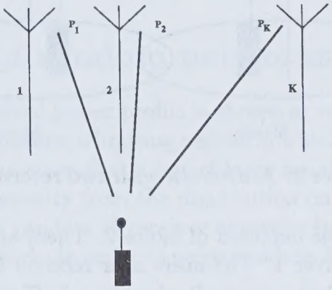


Figure 1: Diversity reception for a single user with K receivers

2.1 Multi-user theory

Multi-user information theory is a generalisation of Shannon's theory and is outlined in [2], [3]. There, M users share a common bandwidth $-W \cdots W$ and send to a common receiver in the presence of external noise of spectral density $\eta/2$. If user i requires bit rate R_i and is received at power P_i then the capacity region is given by the constraints

$$\sum_{i \in \mathcal{L}} R_i \leq W \log_2 \left(1 + \frac{1}{\eta W} \sum_{i \in \mathcal{L}} P_i \right)$$

where $\mathcal{L}$ ranges over all subsets of $\{1, \cdots, M\}$.

2.2 Multi-user, multi-receiver

The extension to the K receiver case is as follows. The signal of user $i = 1 \cdots M$ is received at receiver $j = 1 \cdots K$ at power $P_{i,j}$. $\mathcal{P}_i$ is defined by replacing P_j by $P_{i,j}$ in (2) so that $\mathcal{P}_i$ is the power profile for the i'th user. Then the capacity region is

$$\sum_{i \in \mathcal{L}} R_i \leq W \log_2 \left(|\, I + \frac{1}{\eta W} \sum_{i \in \mathcal{L}} \mathcal{P}_i \,| \right) \tag{4}$$

where, again, $\mathcal{L}$ ranges over subsets of $\{1, \cdots, M\}$. It is also shown in [6] that when $K = 2$

$$\sum_{i\in\mathcal{L}} R_i \le W \log_2 \left(1 + \sum_{i\in\mathcal{L}} \frac{P_{i,1} + P_{i,2}}{\eta W} + \sum_{i\in\mathcal{L}} \sum_{\substack{j > i \\ j\in\mathcal{L}}} \left(\frac{\sqrt{P_{i,1}P_{j,2}} - \sqrt{P_{i,2}P_{j,1}}}{\eta W} \right)^2 \right) \tag{5}$$

2.3 Example

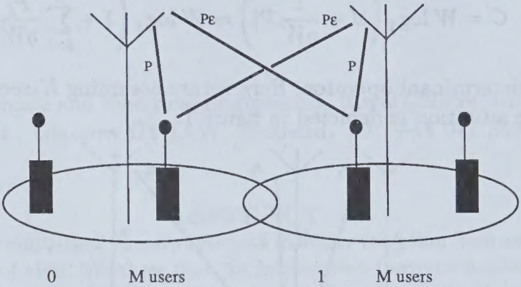


Figure 2: A network with two receivers

Consider the two cell example depicted in figure 2. There are M users close to receiver 0 and M users close to receiver 1. The users near receiver 0 are received at receiver 0 at power P and at receiver 1 at power εP where $\varepsilon < 1$. Similarly, those near receiver 1 are received at 1 at power P and at 0 at power εP. We shall adopt terminology from mobile radio and say that the M users near receiver 0 are in cell 0 and that those near receiver 1 are in cell 1. All users require the same bit rate R.

Theorem 1 *The capacity of the two cell network is given by*

$$C = \frac{W}{2M} \log_2 \left(1 + \frac{2M(1+\varepsilon)P}{\eta W} + \frac{M^2(1-\varepsilon)^2 P^2}{(\eta W)^2} \right)$$

and this can be achieved when the full bandwidth W is used in each cell with orthogonality within cells.

Proof

By (5), the $2M$-user constraint is given by

$$2MR \le W \log_2 \left(1 + \frac{2M(1+\varepsilon)P}{\eta W} + \frac{M^2(1-\varepsilon)^2 P^2}{(\eta W)^2} \right)$$

We show that this is the tightest constraint by exhibiting a transmission scheme that has capacity given by this constraint. First, consider the case $M = 1$; just one user in each cell. Then the capacity is

$$\frac{W}{2} \log_2 \left(1 + \frac{2(1+\varepsilon)P}{\eta W} + \frac{(1-\varepsilon)^2 P^2}{(\eta W)^2} \right)$$

Now consider $M > 1$. Suppose we use the full bandwidth in each cell but partition it equally amongst the M users. Then each user has allocated a bandwidth $\frac{W}{M}$ and has a single interferer from the other cell occupying the same bandwidth. The $M = 1$ case above shows that the capacity of such a subchannel is

$$\frac{W}{2M}\log_2\left(1+\frac{2M(1+\varepsilon)P}{\eta W}+\frac{M^2(1-\varepsilon)^2P^2}{(\eta W)^2}\right)$$

and hence is optimal. Thus we have the intuitively reasonable result that the capacity can be achieved by partitioning bandwidth within cells (where signals are received at the same power) but not partitioning between cells. In the spirit of information theory we do not explicitly give a coding and decoding algorithm here! •

Note that information theory not only gives the capacity but also insights into how it may be achieved. The result of theorem 1 can be generalised to a wide class of symmetrical cellular networks. The above argument does not rely on the final constraint being the most onerous.

3 A sequence of random networks

In this section a user's received power profile is chosen at random according to a probability distribution P. We consider a limiting regime in which the bandwidth and number of users increase in step and show that a law of large numbers operates so that we can find the limiting bit-rate capacity from the distribution on the power profiles.

We define a sequence of random K receiver networks indexed by M where M is the number of users in the network. In order to accommodate M users we let the bandwidth W grow linearly with M:

$$W(M) \equiv W_0 M$$

for some bandwidth per user W_0. Let $\{1, \cdots, n\}$ label the set of n distinct power profiles $\mathcal{P} \equiv \{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$. We assume that a user is allocated one of these according to the probability mass function $p_i \quad i = 1 \cdots n$.

Given the Mth random network we can compute the set of constraints that define the capacity of this network. There is a constraint corresponding to each subset of users but this can be simplified as there are only n possible K-tuples of received powers. Thus for each constraint it is enough to know the numbers of users corresponding to each value in $\mathcal{P}$. Let us place these numbers for a particular constraint into the n-vector $J^{(M)}$. Then the constraint is given by

$$R \leq \frac{W(M)}{\sum_{i=1}^n J_i^{(M)}}\log_2\left(\mid I+\sum_{i=1}^n\frac{J_i^{(M)}\mathcal{P}_i}{\eta W(M)}\mid\right)$$

It is useful in the limiting process to rewrite this in terms of proportions. Let $q^{(M)} = \frac{1}{M}J^{(M)}$ so that $q_i^{(M)}$ is the proportion of the M users that have value $\mathcal{P}_i$ and are in the subset of users corresponding to the constraint. Then we can rewrite the above as

$$R \leq f^{(M)}$$

where

$$f^{(M)} \equiv \frac{W_0}{\sum_{i=1}^n q_i^{(M)}}\log_2\left(\mid I+\sum_{i=1}^n\frac{q_i^{(M)}\mathcal{P}_i}{\eta W_0}\mid\right)$$

There are $2^M - 1$ constraints so as M grows it becomes computationally intractable to compute the tightest constraint. Fortunately, we can use the law of large numbers to find the value of the tightest constraint (the capacity) in the limit as M tends to infinity.

Define the set θ by

$$\theta \equiv \{q : q \leq p\}$$

and let θ^+ be the elements of θ with at least one component greater than 0. Define the value $\tilde{f}$ by

$$\tilde{f} \equiv inf\{f(q) : q \in \theta^+\}$$

Then the following theorem holds.

Theorem 2 *Let $\tilde{f}^{(M)}$ be the value of the tightest constraint in the Mth network. Then with probability* 1

$$lim_{M\uparrow\infty}\tilde{f}^{(M)} = \tilde{f}$$

Proof

This is proved rigorously in [6], so we just summarise that proof here. We let $\tilde{q} \in \theta^+$ be a minimum point of f. By the law of large numbers we can find, with probability 1, a sequence of constraints $J^{(M)}$ with corresponding proportions $q^{(M)}$ such that

$$q^{(M)} \rightarrow \tilde{q}$$

Indeed, this is true for any element of θ.

Now let $r^{(M)}$ be the proportions corresponding to the tightest constraint in the Mth network. Again by the law of large numbers, limit points of $r^{(M)}$ must exist and be elements of θ. Let $\tilde{r} \in \theta$ be such a limit point. But

$$f(r^M) \leq f(q^M)$$

and $\inf f$ is not attained in a neighbourhood of zero ([6]), so the sequence (r^M) is bounded away from 0 in the limit. By the continuity of f away from 0 there is a subsequence N_n such that

$$f(r^{N_n}) \rightarrow f(\tilde{r})$$

and so,

$$f(\tilde{r}) \leq f(\tilde{q}) = \tilde{f}$$

from which it follows that $f(\tilde{r}) = \tilde{f}$. Thus all limit points of $r^{(M)}$ have the value $\tilde{f}$ from which it follows that

$$f(r^{(M)}) \rightarrow \tilde{f} \quad \bullet$$

A simplifying feature of the problem of minimising f over θ arises from the convexity of θ and the special nature of f. It is shown in [5] that a minimum point of f on θ^+ must occur at an extreme point of θ. But the set of extreme points of θ is

$$\{\mathbf{q} : \quad q_i = 0 \quad \text{or} \quad p_i \quad i = 1 \cdots n\}$$

and we arrive at the following result:

Theorem 3 *In the limiting regime $M \uparrow \infty$, there are $2^n - 1$ constraints on R and these are*

$$R \leq \frac{W_0}{\sum_{i=1}^{n} q_i} \log_2 \left(| I + \sum_{i=1}^{n} \frac{q_i \mathcal{P}_i}{\eta W_0} | \right)$$

where

$$q_i = 0 \quad or \quad p_i \qquad i = 1 \cdots n \qquad \mathbf{q} \neq 0$$

Thus there is an inherent simplification as the system becomes deterministic. Rather than having to check $2^M - 1$ constraints as M gets large, we only need to check $2^n - 1$ where n is the number of possible power profiles. Of course, n may be extremely large (the continuous case is examined in [6]), but one might hope that a discrete model in which n is not so large may still be a good approximation.

4 Bandwidth partitioning

We now consider some insights into how spectrum should be allocated in mobile radio that can be deduced from information theory. We assume that all users require the same bit rate R bits/sec.

Our first point is that spread spectrum ([4], [7]) can achieve capacity. Let the bandwidth W be partitioned into J subchannels, the j'th of bandwidth $\alpha_j W$ where $\sum_{j=1}^{J} \alpha_j = 1$. Consider the following "spread spectrum" scheme. Each user requires rate $\alpha_j R$ from subchannel j and puts a proportion α_j of its power profile in subchannel j. To see that this approach can achieve capacity note that for any group $\mathcal{L}$ of users the corresponding constraint in the j'th subchannel is

$$\alpha_j(| \mathcal{L} | R) \leq \alpha_j W \log_2 \left(| I + \frac{1}{\eta \alpha_j W} \sum_{l \in \mathcal{L}} \alpha_j \mathcal{P}_l | \right)$$

or

$$| \mathcal{L} | R \leq W \log_2 \left(| I + \frac{1}{\eta W} \sum_{l \in \mathcal{L}} \mathcal{P}_l | \right)$$

Thus each subchannel has a proportion α_j of the total channel capacity.

This result does not imply that it is necessarily suboptimal to partition users into subchannels. Indeed, it is shown in [6] that it is optimal to partition users if their power profiles are identical. Conversely, consider the following theorem, which gives necessary conditions for it to be optimal to partition groups of users into orthogonal subchannels. In [6] we deal with a fixed network, but here we adapt the proof given there to the sequence of random networks considered in the previous section. Let I be a set of power profiles corresponding to a tightest constraint in the set of limiting constraints.

Theorem 4 *Let the bandwidth W be partitioned into J subchannels, the j'th of bandwidth $\alpha_j W$ where $\sum_{j=1}^{J} \alpha_j = 1$. Partition the power profiles in the tightest constraint set I into J groups of power profiles $\mathcal{G}_1 \cdots \mathcal{G}_J$ and users in group $\mathcal{G}_j$ are allocated to the j'th subchannel. A user is allocated to power profile k with probability p_k and we consider the limiting regime of section 3 in which the number of users M in the network tends* *to infinity. If this partitioning achieves capacity (in the limiting sense) then*

$$\alpha_j = \frac{\sum_{k\in\mathcal{G}_j} p_k}{\sum_{k\in I} p_k}$$

$$\frac{1}{\alpha_j}\sum_{k\in\mathcal{G}_j} p_k\mathcal{P}_k = \sum_{k\in I} p_k\mathcal{P}_k$$

Proof

Suppose group $\mathcal{G}_j$ is allocated subchannel j for $j = 1\cdots J$. By Theorem 2 we find that an asymptotic constraint imposed on R by the j'th subchannel is given by

$$\begin{aligned}\sum_{k\in\mathcal{G}_j} p_k R &\le \alpha_j W_0 \log_2\left(| I + \frac{1}{\eta\alpha_j W_0}\sum_{k\in\mathcal{G}_j} p_k\mathcal{P}_k | \right)\\ &= \alpha_j f(A_j)\end{aligned}$$

where

$$\begin{aligned}f(A) &\equiv W_0\log_2\left(| I + \frac{1}{\eta W_0} A | \right)\\ A_j &\equiv \frac{1}{\alpha_j}\sum_{k\in\mathcal{G}_j} p_k\mathcal{P}_k\end{aligned}$$

and thus

$$R \le \frac{1}{\sum_{k\in I} p_k}\sum_{j=1}^{J}\alpha_j f(A_j)$$

By Cover and Thomas [1], or lemma 7 in the appendix to [6], f is strictly concave and so

$$\begin{aligned}\frac{1}{\sum_{k\in I} p_k}\sum_{j=1}^{J}\alpha_j f(A_j) &\le \frac{1}{\sum_{k\in I} p_k} f(\sum_{j=1}^{J}\alpha_j A_j) \qquad (6)\\ &= \frac{W_0}{\sum_{k\in I} p_k}\log_2\left(| I + \frac{1}{\eta W_0}\sum_{k\in I} p_k\mathcal{P}_k | \right)\end{aligned}$$

and there is equality if and only if

$$A_j = A_1 \quad j = 1\cdots J$$

Thus if the above scheme achieves capacity then

$$\frac{1}{\alpha_j}\sum_{k\in\mathcal{G}_j} p_k\mathcal{P}_k = \frac{1}{\alpha_1}\sum_{k\in\mathcal{G}_1} p_k\mathcal{P}_k \quad j = 1\cdots J.$$

Also, since each subchannel is required to have the same capacity,

$$\frac{\alpha_j W_0}{\sum_{k\in\mathcal{G}_j} p_k}\log_2\left(| I + \frac{1}{\eta\alpha_j W_0}\sum_{k\in\mathcal{G}_j} p_k\mathcal{P}_k | \right) = \frac{\alpha_1 W_0}{\sum_{k\in\mathcal{G}_1} p_k}\log_2\left(| I + \frac{1}{\eta\alpha_1 W_0}\sum_{k\in\mathcal{G}_1} p_k\mathcal{P}_k | \right)$$

and thus $\alpha_j = \frac{\sum_{k\in\mathcal{G}_j} p_k}{\sum_{k\in I} p_k} \quad j = 1\cdots J.$ •

Corollary 1 *If each group $\mathcal{G}_j$ consists of a single power profile and if capacity is achieved then*

$$\alpha_j = \frac{p_j}{\sum_{k \in I} p_k}$$

and

$$\mathcal{P}_j = \mathcal{P}_1 \quad j = 1 \cdots J$$

Note that the same arguments go through in a fixed network ([6]) so that the result does not just hold in a limiting sense. We conclude that it is only optimal to partition users into subchannels if the average power profiles in each group are the same. In a cellular system with reuse partitioning, groups corresponding to adjacent cells are partitioned in bandwidth by the reuse rule, but their average power profiles are not the same. Theorem 4 shows that this arrangement is suboptimal. But how great is this difference?

Consider the symmetrical two cell model of section 2.3. Define W_0 by $W_0 \equiv W/M$ so that W_0 is the per user bandwidth. Then we have seen that the capacity is given by

$$C = \frac{W_0}{2} \log_2 \left(1 + \frac{2(1+\varepsilon)P}{\eta W_0} + \frac{(1-\varepsilon)^2 P^2}{(\eta W_0)^2} \right) \tag{7}$$

and that this capacity can be achieved when the full bandwidth W is used in each cell with orthogonality within cells. With bandwidth partitioning of cells (necessary in practice in FDMA systems because of interference fluctuations) the capacity is given by

$$C_{fdma} = \frac{W_0}{2} \log_2 \left(1 + \frac{2P}{\eta W_0}(1+\varepsilon) \right) \tag{8}$$

We expect that bandwidth will always be a scarce resource so that W_0 will be small. In the limit as $\frac{P}{\eta W_0} \uparrow \infty$ the ratio of (7) to (8) converges to 2, capacity is halved. In general, we might expect the capacity of an FDMA system will be $\frac{1}{r}C$ (in a limiting sense) where r is the reuse factor (r can be 7 or even higher in existing cellular networks). However, our model does not take into account the interference from users reusing the same frequency in co-channel cells. This is because we have only considered the two cell case here. Co-channel interference comes from a small number of users and cannot be modelled by white noise. For this reason we cannot apply information theory to reuse networks in the way we have done in (8). Indeed, we expect that the FDMA capacity is even smaller in these reuse networks.

5 Economies of scale

Consider again our sequence of random networks. In the M'th network, the tightest constraint $f(r^M)$ differs from the limit, $\tilde{f}$, by a random quantity of $O\left(\frac{1}{\sqrt{M}}\right)$, for M large. To see this consider a network with a proportion $\tilde{q}$ minimising f over θ. Let $r^{(M)}$ be the proportion corresponding to the tightest constraint in the Mth network, as above. By the central limit theorem, the set of empirical proportions that correspond to possible constraints lies in $\theta^{(M)}$ where

$$\theta^{(M)} \equiv \{q : q \le p + \frac{1}{\sqrt{M}} \xi^{(M)}\}$$

and $\xi^{(M)} \Rightarrow N(\mathbf{0}, \Sigma)$ for some singular covariance matrix Σ. Let $q_\theta^{(M)}$ be a value which minimises f over $\theta^{(M)}$ and is bounded away from 0 in the limit. Then by the continuity of f

$$\mid f(r^{(M)}) - f(q_\theta^{(M)}) \mid = O\left(\frac{1}{M}\right)$$

and

$$\mid f(q_\theta^{(M)}) - f(\tilde{q}) \mid = O\left(\frac{1}{\sqrt{M}}\right)$$

Therefore

$$\mid f(r^{(M)}) - f(\tilde{q}) \mid = O\left(\frac{1}{\sqrt{M}}\right)$$

In mobile radio, system capacity must be set aside to deal with fluctuations in signal strength. From an information-theoretic point of view the standard deviation of the interference goes up on a scale of $O(\sqrt{M})$ and this is shared over M users making the per-user effect $O\left(\frac{1}{\sqrt{M}}\right)$. Thus the fraction of system capacity needed to be set aside to allow for these fluctuations decreases as the scale of the system increases: a ten fold increase in bandwidth can lead to significantly more than a ten fold increase in capacity. For efficiency, one would like this to be a property of practical transmission schemes. This $O\left(\frac{1}{\sqrt{M}}\right)$ effect arises from averaging, and spread spectrum is a means of multiplexing users onto one channel so that this averaging can take place. It is not difficult to show that the per-user interference effects in spread spectrum are $O\left(\frac{1}{\sqrt{M}}\right)$ (from the central limit theorem). In contrast, the interference effects are $O(1)$ in an FDMA/reuse system.

6 Conclusions

We have shown that the information-theoretic capacity of a random network is well defined. The notion of radio network capacity is not an unstable concept changing significantly with changes in user configuration but becomes deterministic as the scale of the system increases. The capacity can be used to measure the efficiency of practical transmission schemes.

We have also used information theory to draw insights into how spectrum should be allocated in mobile radio. In FDMA systems operating under the reuse rule, capacity is lost precisely because only a fraction of the bandwidth can be used in each cell. In spread spectrum, the full bandwidth can be used in each cell. In fact, we show that capacity can be approached in the class of spread spectrum schemes. The economies of scale analysis shows, in another way, that spread spectrum is a requirement of efficient transmission schemes.

Acknowledgements The authors are very grateful to Frank Kelly, Professor of the mathematics of systems at Cambridge university, for many useful discussions.

References

[1] Cover, T., Thomas, J. (1988) Determinant Inequalities via Information Theory *SIAM J. Matrix Anal. Appl.* Vol. 9 No. 3 July:384–392

[2] Gallager, R. G. (1985) A perspective on Multiaccess Channels. *IEEE Transactions on Information Theory* IT-31 No.2 :124–142

[3] Gamal, A. E., Cover, T. M. (1980) Multiple User Information Theory. *Proceedings of the IEEE* 68:12:1466–1483

[4] Gilhousen, K. S., Jacobs, I. M., Padovani, R., Viterbi, A. J., Weaver, L. A., Wheatley, C. E. (1990) On the capacity of a cellular CDMA system *I.E.E.E. Transactions on Vehicular Technology Vol. 40 No. 2 May 1991.*:303–312

[5] Hanly, S.V. (1993) Capacity of multi-receiver radio networks *PhD thesis, in preparation*

[6] Hanly S.V. , Whiting P.A. (1991) Information-theoretic capacity of multi-receiver networks *to appear in Telecommunications Systems*

[7] Pickholtz R. L., Schilling D. L., Milstein L. B. (1982) Theory of Spread-Spectrum Communications-A Tutorial *IEEE Transactions on Communications* Com-30:5:855–884OB

[8] Shannon, C. E. (1948) A Mathematical Theory of Communication *Bell System Technical Journal* Vol. 27:379–423

A Physically Based Approach to Information Theory for Thin Film Magnetic Recording

Joseph A. O'Sullivan
Donald G. Porter
Ronald S. Indeck
Marcel W. Muller
Department of Electrical Engineering
Washington University
St. Louis, MO 63130

ABSTRACT

Conventional approaches to the study of information theory for magnetic recording use relatively simple models for the magnetic medium which do not account for many interference sources. Typically these models account for a few types of noise sources, ignoring many others. Most models also assume Gaussian statistics for the noise sources. In this paper, we propose a more complicated model for thin film magnetic media and show preliminary analysis to demonstrate its ability to model physical properties of media. Some initial ramifications of using this model (or other advanced models) for studying the information theory of magnetic recording are explored. A major result of this paper is the development of a new methodology for computing the information capacity of magnetic media in bits per unit area. This computation is independent of the method of storing information and thus is a fundamental limit for any magnetic recording system. This represents progress in unifying physical descriptions of magnetic media with the study of the information capacity of magnetic information storage systems.

1. Introduction

This paper introduces a computationally tractable, physically based model for magnetic media intended for use in studying information theoretic properties of magnetic storage systems. The proposed model is most appropriate for thin film and particulate media. Thin film media are briefly discussed in Section 2. The model is discussed in Section 3 and is essentially a spin glass model [1,2,3]. The connection of spin glass models to neural networks [2,3] has proven worthwhile in the study of the capacity of neural networks. It is our intention to exploit these same methods to study the capacity of magnetic storage systems using our model. It is important to emphasize that the starting point of our study of capacity is to look at the capacity per unit area on the surface of the magnetic medium. This then implies fundamental limits on the capacity of magnetic storage systems. Thus the study of information theory from the input-output point of view is connected to a study of information theory at the surface of the actual storage medium.

Several models for magnetic systems are discussed in [4]. These models include the runlength limited channel model, models which account directly for bit-shift [5], and a

---

This work was supported in part by the National Science Foundation under grants ECS-8957148 (PYI) and ECS-9100157.

runlength-limited channel model with noisy transition measurements [6]. Other models account for crosstalk and intersymbol interference [7,8,9]. Several innovative ways to adaptively equalize these sources of interference have been proposed [10,11]. These are input-output models which do not describe the actual physics. They model the manner in which the channel affects the information bits and are limited. We anticipate that the model introduced here will allow for more precise computations of the capacity of magnetic systems and may lead to improved read/write techniques.

A more sophisticated model which has appeared in the literature is the French-Wolf model [12]. McLaughlin and Neuhoff have improved on the French-Wolf model [13,14] and have used their improved model to compute bounds on the capacity of magnetic information channels and to study source coding for analog data for these channels. This model accounts for media noise, electronics noise, filtering in the read process, and the variation of those quantities as the track width and the linear density of bits vary. While their model has proven very useful and some of their results are very impressive, their model still has limitations. One limitation is the assumption of Gaussian noise. A second limitation is the inability of the model to account for bit shift due to magnetic variations at the surface of the medium. A third limitation is its inability to include crosstalk.

Our model will include these other affects. To include other affects, our model must be more sophisticated. The main limitation of our model is its complexity. One area of ongoing research is the simplification of this model in special situations.

The model proposed may be thought of as a simplification of models used to study the physics of magnetics media. Models such as those discussed by Zhu and Bertram [15,16,17,18] and Mansuripur and Giles [19,20] simulate the solution of the Landau-Lifshitz-Gilbert differential equation. To our knowledge, no attempt has been made to use these very complicated models for studying signal processing properties of magnetic information systems. These models do take into account a wide variety of phenomena, however. The model presented below also takes into account these phenomena, but does not attempt to solve the Landau-Lifshitz-Gilbert equation. The model has discrete jumps in magnetic moments of individual grains. Since it is at the level of grains, it is very detailed. Some reduction in complexity is achieved by allowing only a finite number of orientations of magnetic moments of these grains.

Another major goal of this work is to bring together research in the physics of magnetic domains and research in signal processing for magnetic information systems. By developing a model for magnetic recording of use to both, interaction may be enhanced.

2. Thin Film Magnetic Media

This section presents a brief high-level description of thin film magnetic media, from the point of view our model represents. This is not meant to be a detailed physical description.

Thin film media are deposited on substrates in such a manner that independent crystallites initiate growth at various sites across the surface. As the crystals grow, they merge with other crystals at irregular interfaces. Depending on the material and the orientations of the individual crystals, this interface may or may not be abrupt. All crystals are formed independently and have random orientations. At the end of the deposition process, the crystals have grown into what are usually called grains. Media are often characterized by their granularity: the typical sizes of, orientations of, and interactions between the grains. At one extreme are particulate media which are composed of physically independent grains which interact only through dipolar effects. Many media have a tendency for adjacent grains to have magnetic moments which are aligned. The strength of this exchange interaction is a parameter in our model.

In order to get a sense for the order of magnitude of the grains under discussion, our model assumes grains are about as wide as they are thick. The media under consideration may have thicknesses of a fraction of a micron.

3. Magnetic Model

Our model assumes grains are located on sites on a rectangular two dimensional lattice. The grains are assumed not to overlap. The magnetic moments of individual grains are assumed to lie in the plane of the lattice. Several types of interactions between grains are included in the model, and several sources of randomness are included.

Each grain has an axis of anisotropy (the easy axis), $\mathbf{u}_j$, in the plane of the lattice. These axes are determined by independent, identically distributed random angles, θ_j (due to the random nature of crystal growth). To make the model computationally more tractable, we assume that the anisotropy is so strong that the magnetic moment of an individual grain lies along its easy axis. Thus, for each grain a sign, $s_j \in \{-1,1\}$, determines the magnetization which is given by

$$\mathbf{m}_j = m s_j \mathbf{u}_j \ ,$$

where m is the magnitude of the magnetization of each grain.

The grains which are far apart interact through their magnetic fields as dipoles. This dipolar interaction depends on the distance and relative orientation of the grains and is spatially invariant. Note that the magnetic field at a given grain due to a second grain is linear in the magnetic moment of that second grain. The dipolar field at site j is equal to

$$\mathbf{H}_{d,j} = \sum_{l \neq j} \mathbf{A}_{j-l}(\mathbf{m}_l) \ ,$$

where $\mathbf{A}_{j-l}$ is the spatially invariant linear operator representing the dipolar interaction. Given in matrix form for the lattice being in the $x-z$ plane and d_x and d_z being the separation of lattice sites j and l in the x and z coordinates, respectively, we have

$$\mathbf{A}_{j-l} = \frac{K_d}{(d_x^2 + d_z^2)^{5/2}} \begin{bmatrix} 2d_x^2 - d_z^2 & 3d_x d_z \\ 3d_x d_z & 2d_z^2 - d_x^2 \end{bmatrix} ,$$

where K_d is a constant depending on the grain size and the permeability.

Adjacent grains interact additionally through an exchange coupling. This models the tendency the magnetizations of adjacent grains to be parallel. The exchange coupling is modeled by an equivalent magnetic field and is linear in the magnetizations,

$$\mathbf{H}_{ex,j} = \sum_{l \in N(j)} K_{ex,j-l} \mathbf{m}_l \ ,$$

where $K_{ex,j-l}$ is a scalar constant depending on the stiffness coefficient, the saturation magnetization, and the size of the grains. $N(j)$ is the set of neighbors that contribute to the exchange field.

In order to determine in which direction along the easy axis a given grain should point, two steps take place. First, the magnetic field at that grain is computed. This is linear in the magnetic moments of all other grains plus the applied field. Second, this field and the previous magnetic moment of this grain are compared using the Stoner-Wohlfarth model [21]. The result of this comparison determines the direction of the magnetization of that grain.

The Stoner-Wohlfarth model is discussed in more detail in the next section.

Three comments concerning this model are appropriate:

First, other models exist. Some of the most intricate solve the Landau-Lifshitz-Gilbert equation [15,16,17,18,19,20]. Our approach approximates the solution of these nonlinear differential equations. Note that our approach tends to jump to energy minima rather than approach those minima slowly. Our approach is also much less computationally demanding.

Second, the model introduces randomness through the update strategy. A typical use of the model would be the following. A field is applied instantaneously. This means that the previous set of grain magnetizations are no longer valid. Lattice sites are chosen at random and updated by the strategy outlined above. After a large number of lattice site updates, an equilibrium (energy minimum) is declared. Since the order in which the sites are visited is random, the actual energy minimum attained is random.

Third, the model is like a neural network in that a linear combination of the outputs of other sites (their signs) plus a bias (the applied field) is passed through a nonlinear threshold function (from the Stoner-Wohlfarth model) to determine the next value of the sign of the site under consideration. Thus, the literature on the capacity of neural networks (see [22,23,24,25,26]) should prove useful in computing the capacity of our model. The main results from this literature depend on the results in the spin glass literature. While our model has many similarities to typical spin glass models, there is one large difference: the Stoner-Wohlfarth model. Usual spin glass models [1,2] do not include such a switching function to account for coercivity.

4. Stoner-Wohlfarth Model

Stoner and Wohlfarth modeled the energy function for an individual grain as being composed of two terms. The first is an anisotropy energy due to the tendency of the magnetic moment of a grain to align itself with its easy axis. The second term is the energy due to the applied field and represents the tendency of the magnetic moment to align with the applied field. For small applied fields, there are two energy minima. For large applied fields, there is only one energy minimum. A plot of the critical fields at which the transition from two minima to one minimum takes place is known as the Stoner-Wohlfarth astroid. In our model, the Stoner-Wohlfarth astroid is used to determine whether the field at a grain is sufficiently large to cause the magnetic moment to flip. That is, since we assume that the magnetic moment is aligned with the easy axis, the critical field determines when the magnetic moment must flip. For fields lower than critical (inside the astroid), no flip occurs.

5. Information Capacity

There are several ways in which the study of capacity of magnetic media may be carried out based on the model proposed here. The approach taken here is to count the relative logarithmic number of energy minima. That is, the capacity is determined by the logarithm of the number of minima divided by the logarithm of the total number of possible configurations. One limitation of using this method for computing the capacity is that the minima tend not to be isolated. This means that two or more minima may be indistinguishable from a measurement standpoint. The modeling of the measurement process should inlcude this ambiguity.

Results on computing the capacity of media have been obtained and will be published elsewhere. This study is being carried out analytically and computationally. One goal in this study is to motivate results like the following. This result is stated in terms of energy "basins" which may include several energy minima.

Suppose there are N lattice sites and a typical energy basin is chosen. Suppose that, on average, if any k signs are changed on individual grains from any energy minimum in this energy basin, the configuration returns to this energy basin. Then there are approximately $N!/k!(N-k)!$ configurations which are in the "attraction set" for this energy basin. The entropy under such a scenario may be approximated by noticing that the number of basins approximately equals the total number of configurations possible divided by the size of the "attraction set" for each basin. Thus, approximately,

$$|S_{\min}| \approx \frac{2^N}{N!/k!(N-k)!} \approx 2^{N(1-h(k/N))}$$

where $h(p) = -p \log p - (1-p)\log(1-p)$ (this approximation is standard [27,28,29] and is obtained from Sterling's inequality). Then, if it can be shown that k/N converges to $p > 0$ as N gets large, then a bound on the number of minima and hence on the entropy is obtained. Above, the entropy per grain assuming all basins are equally likely equals $1\text{-}h(k/N)$.

6. Conclusions

A physically based model of thin film magnetics is developed and used to study the capacity of magnetic recording systems. It is hoped that this approach will lead to more precise computations of capacity and will yield a closer connection between the study of signal processing strategies for magnetic information systems and the study of the underlying physics. The approach is natural in the sense that it starts with the information storage device and works outward. In the classic communication theory sense, the work discussed here is channel modeling. The model of the channel affects all of the other blocks. Other approaches tend to simplify the channel model or make nonphysical assumptions about the channel in order to be able to use previous results. In our work we will explore the impact of a better channel model on the elements in a magnetic recording system including the read/write heads, channel modulation/demodulation, and channel coding/decoding. Better designs for these other blocks may emerge.

References

1. K. H. Fischer and J. A. Hertz, *Spin Glasses,* Cambridge University Press, Cambridge, 1991.
2. *Statistical Mechanics of Neural Networks,* Springer-Verlag, Berlin, 1990.
3. D. J. Amit, *Modeling Brain Function: The World of Attractor Neural Networks,* Cambridge University Press, Cambridge, 1989.
4. “Special Issue on Coding for Storage Devices,” *IEEE Transactions on Information Theory*, vol. IT-37, May 1991.
5. S. Shamai and E. Zehavi, “Bounds on the Capacity of the Bit-Shift Magnetic Recording Channel,” *IEEE Transactions on Information Theory*, vol. IT-37, pp. 863-872, May 1991.
6. C. D. Heegard, A. Duel-Hallen, and R. Krishnamoorthy, “On the Capacity of the Noisy Runlength Channel,” *IEEE Transactions on Information Theory*, vol. IT-37, pp. 712-720, May 1991.
7. R. W. Wood and D. A. Petersen, “Viterbi Detection of Class IV Partial Response on a Magnetic Recording Channel,” *IEEE Transactions on Communications*, vol. COM-34, pp. 454-461, May 1986.

8. R. Wood, "Enhanced Decision Feedback Equalization," *IEEE Transactions on Magnetics*, vol. MAG-26, pp. 2178-2180, 1990.
9. L. C. Barbosa, "Simultaneous Detection of Readback Signals from Interfering Magnetic Recording Tracks Using Array Heads," *IEEE Transactions on Magnetics*, vol. MAG-26, pp. 2163-2165, September 1990.
10. J. M. Cioffi, W. L. Abbott, H. K. Thapar, C. M. Melas, and K. D. Fisher, "Adaptive Equalization in Magnetic-Disk Storage Channels," *IEEE Communications Magazine*, pp. 14-29, Feb. 1990.
11. W. L. Abbott, J. M. Cioffi, and H. K. Thapar, "Offtrack Interference and Equalization in Magnetic Recording," *IEEE Transactions on Magnetics*, vol. MAG-24, pp. 2964-2966, Nov. 1988.
12. C. A. French and J. K. Wolf, "Bounds on the Capacity of a Peak Power Constrained Gaussian Channel," *IEEE Transactions on Magnetics*, vol. MAG-24, pp. 2247-2262, September 1988.
13. S. W. McLaughlin and D. L. Neuhoff, "Upper Bounds on the Capacity of the Digital Magnetic Recording Channel," *submitted to the IEEE Transactions on Magnetics*, 1992.
14. S. W. McLaughlin and D. L. Neuhoff, "Source-Channel Coding of Analog Data for Digital Magnetic Recording," *submitted to IEEE Transactions on Magnetics*, 1992.
15. J. Zhu and H. N. Bertram, "Recording and Transition Noise Simulation in Thin Film Media," *IEEE Transactions on Magnetics*, vol. MAG-24, pp. 2706-2708, 1988.
16. J. Zhu and H. N. Bertram, "Micromagnetic Studies of Metallic Thin Films," *Journal of Applied Physics*, vol. 63, pp. 3248-3253, 1988.
17. J. Zhu and H. N. Bertram, "Magnetization Reversal in CoCr Thin Films," *Journal of Applied Physics*, vol. 66, pp. 1291-1307, 1989.
18. J. Zhu and H. N. Bertram, "Reversal Mechanisms and Domain Structure in Thin-Film Recording Media," *Journal of Applied Physics*, vol. 69, pp. 6084-6089, 1991.
19. M. Mansuripur and R. Giles, "Simulation of the Magnetization-Reversal Dynamics on the Connection Machine," *Computers in Physics*, vol. 5, pp. 291-302, 1991.
20. R. Giles and M. Mansuripur, "Possible Sources of Coercivity in Thin Films of Amorphous Rare Earth-Transition Metal Alloys," *Computers in Physics*, vol. 5, pp. 204-219, 1991.
21. E. C. Stoner and E. P. Wohlfarth, "Mechanisms of Magnetic Hysteresis in Heterogeneous Alloys," *Proc. R. Soc.*, vol. A240, p. 599, London, 1948.
22. R.J. McEliece, E.C. Posner, E.R. Rodemich, and S.S. Venkatesh, "The Capacity of the Hopfield Associative Memory," *IEEE Transactions of Information Theory*, vol. IT-33, pp. 461-482, July 1987.
23. S.S. Venkatesh and D. Psaltis, "Linear and Logarithmic Capacities in Associative Neural Networks," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 558-568, May 1989.
24. A. Kuh and B. W. Dickinson, "Information Capacity of Associative Memories," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 59-68, Jan. 1989.
25. H. J. Sussman, "On the Number of Memories that can be Perfectly Stored in a Neural Net with Hebb Weights," *IEEE Transactions on Information Theory*, vol. IT-35, pp. 174-178, Jan. 1989.

26. Y. Baram, "On the Capacity of Ternary Hebbian Networks," *IEEE Transactions on Information Theory*, vol. IT-37, pp. 528-534, May 1991.

27. R. Blahut, *Principles and Practice of Information Theory,* Addison Wesley, Reading, MA, 1987.

28. T. M. Cover and J. A. Thomas, *Elements of Information Theory,* Wiley, New York, 1991.

29. R. S. Ellis, *Entropy, Large Deviations, and Statistical Mechanics,* Springer-Verlag, New York, 1985.

On Generating Random Bits from an Arbitrary Random Source*

Sridhar Vembu *Sergio Verdú*
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

Abstract

We address the following question: Suppose we are given a random source and want to use it as a random number generator, how many approximately fair bits can we generate from it? We answer this question with two different measures of approximation between probability distributions: the variational distance and the normalized divergence. The results of this paper provide an operational characterization of the *inf-entropy rate* of a source, defined in Han and Verdú, [1]. We also relate the inf and sup entropy rates to the quantity called *resolvability* of channels, also defined in [1].

1 Introduction

The recent results of Han and Verdú, [1], have opened some new questions in information theory. We focus on the source coding results of [1] in this article. Specifically, we address the following question: suppose we are given a random source and we want to use it as a random number generator, how many fair bits can we generate from it? Of course, we may not be able to get too many "pure" random bits - for example, if the probability masses of the source are irrational. Instead we will focus on synthesizing an "approximately" uniform distribution from the source and address the issue of the largest asymptotic exponential rate at which this can be done. We will call this rate the Intrinsic Randomness rate (IR rate) of the source. In this paper we use two measures of approximation of probability distributions - variational distance and divergence. In both cases the answer to the question turns out to be the same: it will be the *inf-entropy rate* as defined in Han and Verdú, [1]. The results of this article nicely complement the source coding results of [1] where the minimum fixed length encoding rate is shown to be the *sup-entropy rate* for an arbitrary finite alphabet source. Thus we see (Figure 1) that intrinsic randomness plays the counterpart to the minimum source coding rate, analogously to resolvability and channel capacity.

We will also relate the inf-entropy and the sup-entropy rates to the *resolvability* of channels defined in [1].

---

*This work was supported in part by the National Science Foundation under PYI Grant ECSE-8857689, and the Office of Naval Research under Grant N00014-90-J-1734. The work of the first author was also supported by an IBM Graduate Fellowship.

Source	Channel
$\overline{H}(X)$ Fixed length source coding	$\sup_X \overline{I}(X;Y)$ Resolvability
$\underline{H}(X)$ Intrinsic Randomness	$\sup_X \underline{I}(X;Y)$ Channel Capacity

Figure 1: Operational Characterization

Throughout this paper, by source we mean a sequence of finite dimensional distributions $P = \{P^n\}$. Note that we do not impose any consistency requirements - i. e. a higher dimensional distribution need not have the lower dimensional distributions as its marginals. We will denote the finite dimensional random variables by X^n, Y^n etc and their sequences by $\mathbf{X}, \mathbf{Y}$ etc. In a slight abuse of notation, in order to denote the distance between two probability distributions, we use the corresponding random variables. All logarithms in this paper have an arbitrary base greater than 1 and exp refers to that base.

In Cover and Thomas, [2], page 43, problem 7, the question of generating fair coin flips from biased coins is mentioned. The results of this paper could be thought of answering this question in a very general setup, with the relaxation in the requirement of pure coin flips.

In [1] it is shown that the resolvability of a source i. e. the number of bits required in the worst case to approximate the source is shown to be its sup-entropy rate. In this paper we prove that the number of bits we can usefully extract from the source is characterized by its inf-entropy rate.

2 Main Results

2.1 Approximation in Variational Distance

In the spirit of [1] we will adopt the following definitions.

Definition 1 *Given a discrete source* $\mathbf{X}$ *with alphabet* A. *Fix* $\epsilon > 0$. R *is said to be an* ϵ*-achievable Intrinsic Randomness (IR) rate if there exists a sequence of mappings* $\phi_n : A^n \to \{1, 2, 3, \ldots M\}$ *such that for all* $\gamma > 0$ *and for all sufficiently large* n,

$$\frac{\log M}{n} > R - \gamma$$

and

$$d(\phi_n(X^n), U_M) < \epsilon$$

where U_M *is uniformly distributed on* $\{1, 2, 3 \ldots M\}$ *and* d *is the variational distance.*

The variational distance between two probability distributions P_1 and P_2 on an alphabet A is defined to be

$$d(P_1, P_2) = \sum_{\{x \in A\}} |P_1(x) - P_2(x)|$$

By a slight abuse of notation, we will use $d(X, Y)$ to denote the variational distance between the probability distributions that correspond to the random variables X and Y respectively. It is easy to show that

$$d(P_1, P_2) = 2max_{\{E:E\subset A\}}|P_1(E) - P_2(E)|$$

We will find both characterizations of the variational distance useful.

Remark 1: In the above we can use the normalized divergence $\frac{1}{n}D(\phi(X^n)||U_M)$ instead of the variational distance and in order to distinguish this we will call it achievability in the sense of divergence. The results for that approximation measure are proved separately.

Definition 2 *If a rate is ϵ-achievable for all $\epsilon > 0$ then it is called an achievable IR rate.*

Definition 3 *The maximum achievable IR rates is called the MIR rate of the source* $\mathbf{X}$. *We will denote it by* $U(\mathbf{X})$.

Intuitively, the MIR rate refers to the rate of the the maximal uniform distribution we can synthesize from a given source asymptotically.

Now we will define the inf-entropy rate as in [1].

Definition 4 *The inf-entropy rate of* $\mathbf{X}$, $\underline{H}(\mathbf{X})$ *is defined as the largest* α *that satisfies, for each* $\epsilon > 0$,

$$lim_{n\to\infty}P_{X^n}\{x^n \in A^n : \frac{1}{n}\log\frac{1}{P(x^n)} \leq \alpha - \epsilon\} = 0$$

Now we are ready to state our main result: The MIR rate of a source is equal to its inf-entropy rate. We will first prove the converse part and then the direct (achievability) part.

Lemma 1

$$U(\mathbf{X}) \leq \underline{H}(\mathbf{X})$$

where $U(\mathbf{X})$ *is the MIR rate and* $\underline{H}(\mathbf{X})$ *is the inf-entropy rate of* $\mathbf{X}$.

Proof:

By contradiction. Suppose $U > \underline{H}$. Then there exists a $\delta > 0$ such that $U > \underline{H} + 2\delta$. By definition of U it follows that there exists a sequence of mappings $\phi_n : A^n \to \{1, 2, 3, \ldots M\}$ such that for all sufficiently large n,

$$\frac{\log M}{n} > \underline{H} + \delta$$

and

$$d(\phi_n(X^n), U_M) \to 0 \quad \text{as} \quad n \to \infty \tag{1}$$

Now by definition of $\underline{H}$ it follows that

$$P_{X^n}(D) \geq \alpha > 0 \ \text{ infinitely often in } \ n \tag{2}$$

where

$$D = \{x^n \in A^n : \frac{1}{n} \log \frac{1}{P(x^n)} \leq \underline{H} + \frac{\delta}{2}\}$$

We will focus on those blocklengths n for which (2) is true.

It is also clear that the size of D is upper bounded as

$$|D| \leq exp\{n(\underline{H} + \frac{\delta}{2})\}$$

Hence we get, for any mapping ϕ_n,

$$\begin{aligned}
d(\phi_n(X^n), U_M) &\geq P_{X^n}\{\phi_n(X^n) \in \phi_n(D)\} - P_{U_M}(\phi_n(D)) \\
&\geq P_{X^n}(D) - \frac{|\phi_n(D)|}{M} \\
&\geq \alpha - \frac{|\phi_n(D)|}{M} \\
&\geq \alpha - exp\{n(\underline{H} + \frac{\delta}{2})\}exp\{-n(\underline{H} + \delta)\} \\
&= \alpha - exp\{-\frac{n\delta}{2}\} > \frac{\alpha}{2} > 0 \ \text{ infinitely often in } \ n
\end{aligned}$$

where we have used the fact that $|\phi_n(D)| \leq |D| \leq exp\{n(\underline{H} + \frac{\delta}{2})\}$. Hence we get a contradiction of (1) establishing the converse result. ■

Lemma 2

$$U(\mathbf{X}) \geq \underline{H}(\mathbf{X})$$

Proof: We will show that $\underline{H} - 3\gamma$ is an achievable IR rate for any $\gamma > 0$. From the definition of $\underline{H}$ it follows that, for any $\gamma > 0$,

$$\lim_{n \to \infty} P_{X^n}\{x^n : \frac{1}{n} \log \frac{1}{P(x^n)} \leq \underline{H} - \gamma\} = 0$$

Hence the set $B_n = \{x^n : P(x^n) < exp\{-n(\underline{H} - \gamma)\}\}$ is such that $\lim_{n \to \infty} P_{X^n}(B_n) = 1$. So for any $0 < \epsilon < \frac{1}{2}$ and for sufficiently large n,

$$P_{X^n}(B_n) \geq 1 - \epsilon$$

Hence

$$|B_n| \geq (1 - \epsilon)exp\{n(\underline{H} - \gamma)\} \geq \lfloor (1 - \epsilon)exp\{n(\underline{H} - \gamma)\} \rfloor = N \ \ \text{(say)}$$

It is easy to see that, for large n,

$$exp\{n(\underline{H} - 2\gamma)\} \leq N \leq exp\{n(\underline{H} - \gamma)\}$$

We will define a sequence of mappings $\{\phi_n : A^n \to \{1,2,3,\ldots M\}\}$ as follows. For each n, define a partition of A^n in to a collection of sets $\{B(i)\}_{i=1}^M$ as follows. First of all $B(1) = B_n^c$ (complement of B_n). The rest of the $B(i)$'s are defined by the following procedure.

Step 1: $j = 2$;
Step 2: Initialize $B(j) = \{\ \}$.
Step 3: Pick an arbitrary $x^n \in B_n - \bigcup_{i=2}^j B(i)$. Stop if this set is empty.
Step 4: $B(j) \leftarrow B(j) \bigcup \{x^n\}$. We know that $P_{X^n}(x^n) \leq \frac{1}{N}$.
Step 5: Repeat steps 3 and 4 until

$$\frac{n-1}{N} \leq P_{X^n}(B(j)) \leq \frac{n}{N} \tag{3}$$

Step 6: $j \leftarrow j + 1$. Go to step 2.

This procedure results in a partition of B_n in to $\{B(i)\}_2^M$ with each of these subsets of B_n satisfying (3). We call B_n^c as $B(1)$. This procedure basically aggregates elements of B_n in to subsets that have strict bounds on their probability. Now the number of partitions M is bounded as

$$\frac{(1-\epsilon)N}{n} \leq M - 1 \leq \frac{N}{n-1} \tag{4}$$

We now define $\{\phi_n\}$ as:

$$\phi_n(x^n) = i \quad \text{if } x^n \in B(i)$$

Immediately we see that

$$\frac{\log M}{n} \geq \frac{\log N}{n} + \frac{\log(1-\epsilon)}{n} - \frac{\log n}{n} > \underline{H} - 3\gamma \text{ for sufficiently large } n$$

Now the variational distance can be bounded as

$$\begin{aligned} d(\phi_n(X^n), U_M) &\leq P(B(1)) + \frac{1}{M} + \sum_{i=2}^{M} |P(B(i)) - \frac{1}{M}| && (5) \\ &\leq \epsilon + \frac{1}{M} + (M-1)(\frac{n\epsilon}{(1-\epsilon)N} + \frac{1}{N}) && (6) \\ &\leq \epsilon + \frac{n\epsilon}{(n-1)(1-\epsilon)} + \frac{1}{n-1} && (7) \\ &\leq 5\epsilon \text{ for sufficiently large } n && (8) \end{aligned}$$

where (6) follows from (3) and (4). Since the bound on the variational distance is true for any $0 < \epsilon < \frac{1}{2}$, we have established the result. ∎

Remark 2: The above is a statement of the *existence* of a close to uniform distribution on a set of size $N \geq n(\underline{H} - \delta)$ for any $\delta > 0$. From this we can easily show that we can approximate an arbitrary M-type distribution for any $M \leq exp\{n(\underline{H} - 2\delta)\}$. This would involve aggregating the probabilities just as we did in the proof of Lemma 2.

2.2 Approximation in the sense of Normalized Divergence

We will now prove analogous results with normalized divergence rather than variational distance as our measure of approximation of probability distributions. Divergence between two probability distributions P_1 and P_2 is defined as

$$D(P_1||P_2) = \sum_{\{x \in A\}} P_1(x) \log \frac{P_1(x)}{P_2(x)}$$

We will also use the notation $D(X||Y)$ to denote the divergence between the distributions from which X and Y are drawn. Note that divergence is not a metric in the space of probability distributions and is neither stronger nor weaker than the variational distance.

As noted in Remark 1, will replace the variational distance in definition 1 with the normalized divergence $\frac{1}{n}D(\phi(X^n)||U_M)$. Even with this new definition of achievability it turns out that the same result is true, i. e. the largest achievable IR rate is equal to the inf-entropy rate of the source $\underline{H}$.

Before proving this statement we will first explain what the new definition means. Note that

$$D(\phi(X^n)||U_M) = \log M - H(\phi(X^n))$$

where $H(\phi(X^n))$ is the entropy of the M-valued random variable $\phi(X^n)$. Therefore, ϵ-achievability implies that

$$\frac{1}{n} \log M - \epsilon \leq H(\phi(X^n)) \leq \frac{1}{n} \log M$$

where the first inequality is true for large n and the second one is true for any n.

Lemma 3 *The largest achievable IR rate in the sense of divergence of of a source* $\mathbf{X}$*,* $U_d(\mathbf{X})$ *satisfies,* $U_d(\mathbf{X}) \geq \underline{H}(\mathbf{X})$.

Proof:

The proof is by the same aggregation procedure followed in lemma 2 with a slight modification and so will not be repeated here. ∎

The converse part is proved below.

Lemma 4 *If* R *is an achievable IR rate in the sense of divergence, then* $R \leq \underline{H}$.

Proof: The proof is by contradiction. Suppose there is a $R > \underline{H}$ that is achievable. Hence for some $\delta > 0$, $R > \underline{H} + 3\delta$. Let

$$D_n = \{x^n \in A^n : P(x^n) \geq exp\{-n(\underline{H} + \delta)\}\}$$

and

$$P(D_n) = \alpha_n$$

By definition of $\underline{H}$, there exists $\alpha > 0$ such that $\alpha_n \geq \alpha$ infinitely often in n. We will focus on those block lengths n only. The size of D_n is bounded by

$$|D_n| \leq \alpha_n exp\{n(\underline{H} + \delta)\}$$

Let us choose $\epsilon = \frac{\alpha\delta}{4}$. Now achievability of R implies that for we can find a sequence of mappings $\{\phi_n\}$ where $\phi_n : A^n \to \{1, 2, 3, \ldots M\}$ such that for sufficiently large n,

$$\frac{1}{n}\log M > R - \delta \tag{9}$$

and

$$\frac{1}{n}H(\phi_n(X^n)) \geq \frac{\log M}{n} - \epsilon \tag{10}$$

Now we bound the entropy of $H(\phi_n(X^n))$ as follows. Note that $\phi_n(X^n)$ takes values in the set $\{1, 2, \ldots, M\}$. Now define a new transformation ψ_n that takes values in a set of size $e^{n(\underline{H}+\delta)} + M$ as follows: for $x^n \in D_n$, $\psi_n(x^n)$ takes value in the first $e^{n(\underline{H}+\delta)}$ elements and for $x^n \notin D_n$, $\psi_n(x^n)$ takes value in the remaining M elements. It is clear that

$$H(\phi_n(X^n) \leq H(\psi_n(X^n) \tag{11}$$

because we can always construct ϕ from ψ through another deterministic transformation. Now we get

$$\begin{aligned} H(Y_n) &= H(\psi_n(X^n) \\ &= \sum_{i=1}^{e^{n(\underline{H}+\delta)}} P_{Y_n}(i)\log\frac{1}{P_{Y_n}(i)} + \sum_{i=e^{n(\underline{H}+\delta)}+1}^{e^{n(\underline{H}+\delta)}+M} P_{Y_n}(i)\log\frac{1}{P_{Y_n}(i)} \end{aligned}$$

Now $P_{Y_n}(i) \geq e^{-n(\underline{H}+\delta)}$ or $P_{Y_n}(i) = 0$ for the i's in the first term in the right hand side and also the total probability of this set of i's is $P(D_n) = \alpha_n$, and hence the first term in bounded above by $\alpha_n n(\underline{H} + \delta)$. The second term is easily bounded by $(1-\alpha_n)\log M - (1-\alpha_n)\log(1-\alpha_n)$. Putting together all these we obtain,

$$H(\phi_n(X^n)) \leq n\alpha_n(\underline{H} + \delta) + (1-\alpha_n)\log(M-1) - (1-\alpha_n)\log(1-\alpha_n) \tag{12}$$

Now by assumption on R and from (9), we see that $M > exp\{n(\underline{H} + 2\delta)\}$ and using this and (12) we get,

$$\begin{aligned} \frac{1}{n}H(\phi_n(X^n)) &\leq \alpha_n(\frac{\log M}{n} - \delta) + (1-\alpha_n)\frac{\log M}{n} + \frac{(1-\alpha_n)}{n}\log\frac{1}{1-\alpha_n} \\ &\leq \frac{\log M}{n} - \frac{\alpha\delta}{2} \\ &= \frac{\log M}{n} - 2\epsilon \end{aligned}$$

for sufficiently large n, infinitely often in n. This contradicts (10) thereby establishing the result. ∎

The above results giving an operational characterization of $\underline{H}$ complement the fixed length source coding result of [1]. In the finite alphabet case if $\underline{H} = \overline{H}$ then the following limit exists and the above two quantities are also equal to $H = \lim_{n\to\infty}\frac{1}{n}E\log\frac{1}{P(X^n)}$. This is true for example in the stationary, ergodic case. The result presented in the paper may be viewed as a generalization of the asymptotic equipartition property (AEP).

As a nonergodic example, consider independent flips of a coin which is fair with probability $\frac{1}{2}$ and has probability of heads equal to $\frac{1}{\pi}$. Then the minimum achievable source coding rate is 1 bit/symbol, whereas its intrinsic randomness is $h(\frac{1}{\pi})$ and its entropy rate is $\frac{1}{2} + \frac{1}{2}h(\frac{1}{\pi})$, where $h(p) = -p\log p - (1-p)\log(1-p)$.

3 Connection with Resolvability

We will now provide a connection between resolvability and inf-entropy. Resolvability is concerned with the following question: what is the minimum M required so that the if the n-dimensional input distribution is replaced by an M-type, the output distribution is arbitrarily close to the original output distribution in the sense of variational distance. The asymptotic exponential rate of this M, i. e. $\limsup \frac{\log M}{n}$ is called the resolvability of the channel for that particular input. If we supremize over all input distributions we get the resolvability of the channel. In [1], resolvability of a channel is shown to be $S = \sup_{\mathbf{X}} \overline{I}(\mathbf{X}, \mathbf{Y})$ where $\mathbf{X}$ and $\mathbf{Y}$ refer to the channel input and the channel output distributions respectively.

Motivated by the famous source-channel separation theorem in information theory, we ask the following question: we are given a source $\mathcal{S}$ with inf-entropy rate $\underline{H}$. We want to use $\mathcal{S}$ to *emulate* an arbitrary source $\mathcal{S}'$ in the following sense: apply a deterministic transformation on the source $\mathcal{S}$ so that when this is applied to the channel, the output is close in variational distance to that of the source $\mathcal{S}'$. We can show that this is possible if $S < \underline{H}$ and it is not possible if $S > \overline{H}$. If we could show that if $S > \underline{H}$, the source cannot be an approximating source, it would very nicely complement the separation theorem that relates the minimum source coding rate and channel capacity. Unfortunately, we have been unable to answer the question of what happens if $\underline{H} < S < \overline{H}$.

Before we state the result formally we need some definitions. We will denote a sequence of random transformations $\{W^n\}$ by $\mathbf{W}$.

Definition 5 *Suppose we are given a channel* $\mathbf{W}$ *and a source* $\mathbf{Z} = \{Z^n\}$ *with probability law* $\tilde{P} = \{P_{Z^n}\}$. *Fix* $\epsilon > 0$. *We call* $\mathbf{Z}$ *an* $\epsilon-$approximating *source for the channel if for any arbitrary input source* $\mathbf{X}$ *with law* P, *there exists a sequence of mappings* $\psi = \{\psi_n\}$ *with* $\tilde{\mathbf{X}} = \{\psi_n(Z^n)\}$ *having the same alphabet as the input alphabet of the channel such that for sufficiently large* n,

$$d(Y^n, \tilde{Y}^n) < \epsilon$$

where $\mathbf{Y}$ *and* $\tilde{\mathbf{Y}}$ *are the outputs of the channel due to the inputs* $\mathbf{X}$ *and* $\tilde{\mathbf{X}}$ *respectively.*

Definition 6 *We call a source an* approximating *source for a channel if it is* ϵ*-approximating for all* $\epsilon > 0$.

Note that the above definitions do not require an approximating source $\mathbf{Z}$ to have the same alphabet as the channel input alphabet.

Now the connection between resolvability of a channel and and the inf-entropy and sup-enropy rates of a source is provided by the following lemma.

Lemma 5 *A source* $\mathbf{Z}$ *is an approximating source for a channel with resolvability* S *if the inf-entropy rate of the source* $\underline{H}$ *satisfies*

$$\underline{H} > S \tag{13}$$

(ii) If $\overline{H} < S$, *the source cannot be an approximating source for this channel.*

Proof: (Outline) (i) (13) implies that $\underline{H} > S + 2\gamma$ for some $\gamma > 0$. Now by definition of resolvability, we can replace the source $\mathbf{X}$ by a source $\mathbf{X}_{\mathbf{M}}$ whose distribution is of M-type with $M < exp\{n(S+\gamma)\}$ and the output distributions will be close in variational distance. On the other hand, by the Remark following Lemma 2, we can approximate any N type distribution using the given source distribution of $\mathbf{Z}$ if $N < exp\{n(\underline{H} - \gamma)\}$. Since $\underline{H} > S + 2\gamma$, we see immediately that $M < exp\{n(\underline{H} - \gamma)\}$. Now the result will follow from noting that if two input distributions are close in variational distance, the corresponding channel output distributions will be close as well.

(ii) Assume the converse, i. e. $\overline{H} + 2\delta < S$ for some $\delta > 0$ and it is an approximating source for the channel. Now a source with sup-entropy rate $\overline{H}$ can be approximated arbitrarily closely in the sense of variational distance by a M-type distribution where $M = exp\{n(\overline{H} + \delta)\}$ when n is sufficiently large. Hence this approximating source can be replaced by an M-type source which is also an approximating source where $\frac{\log M}{n} < S - \delta$ and this will contradict the definition of resolvability establishing the desired result.

∎

4 Conclusion

In this paper, we have addressed the question of using an arbitrary random source as a random number generator. This gives an operational characterization to the inf-entropy rate. This complements the characterization of the sup-entropy rate found in [1]. Some further directions in this area could be to extend these results in a rate distortion framework. Also our results leave a gap in the connection between resolvability and inf and sup entropy rates. It would be interesting to find out if this gap can be bridged.

Another interesting problem to be considered is to extend these and the results of [1] with another measure of distance between probability distributions called the $\overline{d}$ distance introduced by Ornstein and discussed in [3], [4]. In [5], it is shown that the $\overline{d}$ distance is over bounded by half the variational distance. Hence the direct part of the results hold for the $\overline{d}$ distance. We recently have succeeded in proving the converse part as well and this will be presented in a subsequent paper.

References

[1] T. S. Han and S. Verdú, "Approximation Theory of Output Statistics," submitted for publication. Also, presented at the *1992 IEEE Information Theory Workshop,* Salvador, Brazil, June 1992.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, pp. 43, problem 7. John Wiley & sons, 1991.

[3] D. S. Ornstein, "An application of ergodic theory to probability theory," *Annals of Probability*, vol. I, pp. 43-65, 1973.

[4] R. M. Gray, D. L. Neuhoff, P. C. Shields, " A generalization of Ornstein's distance with applications to Information Theory," *Annals of Probability*, vol. III, pp. 315-328, April 1975.

[5] R. M. Gray, D. S. Ornstein, "Block coding for discrete stationary d-continuous noisy channels," *IEEE Transactions on Information Theory*, **IT-25**, pp. 292-306, 1979.

SOURCE CODING THEOREM AND ITS CONVERSE FOR DISCRETE TIME MS PERIODIC GAUSSIAN SOURCES

Wen-Jyi Hwang and Haluk Derin
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003

Abstract

For a class of nonstationary MS (mean-squared) periodic Gaussian sources and mean-squared error distortion measure, the source coding theorem and its converse are proved. For the proof of the theorems, we define a new rate distortion function $R(D)$ for these nonstationary sources and prove its uniqueness and existence. We show the existence of optimal code using the transform coding scheme. In addition, by proving that $R(D) = 0$, we obtain the converse coding theorem which states that there exists no code with rate less than $R(D)$. Finally, with the proof of the source coding theorem and its converse, it is demonstrated that the newly defined rate distortion function for MS periodic sources is consistent with the usual concept of a rate distortion function as defined for stationary sources.

1 Introduction

Source coding theory provides theoretical bounds for the performance of all coding schemes. In order to evaluate the performance of any coding scheme, it is necessary to know what these bounds are. The knowledge of these bounds also provides useful information on the potential existence of more complex quantizers that perform closer to the bound than the existing ones.

It is well known [1] [2] that, for stationary and ergodic sources with the single-letter based distortion measure and any distortion $D > 0$, there exists a code with a rate arbitrarily close to $R(D)$ and distortion arbitrarily close to D, where $R(D)$ is the rate distortion function [1] [2]. For stationary Gaussian sources, it has been found that a similar source coding theorem exists for some classes of nonergodic sources as well. Gallager [1] proves the existence of an optimal code for the stationary continuous time Gaussian sources with integrable and bounded power spectra and with the mean-squared error distortion measure. Under the same or more restrictive constraints on the power spectrum, source coding theorems for tree and trellis codes are proved in [3], [4], [5]. Recently, Hwang and Derin [6] relaxed the bounded power spectrum condition, which is a common assumption in [1], [3], [4], [5], and showed the existence of an optimal code for discrete time stationary Gaussian sources with integrable power spectra.

In this paper, we present and prove the source coding theorem for MS periodic discrete time Gaussian sources, which are a class of nonstationary sources discussed in [7]. Due to the nonstationarity of the sources, we define a new rate distortion function, $R(D)$, for those sources and prove its existence and uniqueness using techniques similar to the one used by Gallager [1] for stationary sources. The proof of the existence of codes with rates arbitrarily close to the $R(D)$ for MS periodic Gaussian sources uses the transform coding method similar to that of [1], [3], [4], [6], in that, the code is operative in the transform domain, except for the fact that the sources are nonstationary. To prove the converse coding theorem, which states the nonexistence of the code with rate less than $R(D)$ for any distortion $D > 0$, first we show that $R(D) = 0$ for any $D > 0$, and the theorem therefore follows by the fact that there exists no code with negative rate. Finally, based on these two theorems, we conclude that the $R(D)$ we defined for MS periodic Gaussian sources represents the minimum rate which can be achieved for these sources, which is consistent with the usual concept of a rate distortion function as defined for stationary sources.

The remainder of this paper is organized as follows: in Section 2, we review some basic results in rate distortion theory and MS periodic processes. Section 3 presents the definition of rate distortion function for MS periodic sources and shows its uniqueness and existence. Section 4 then presents the source coding theorem for MS periodic Gaussian sources. In Section 5, we present and prove the converse coding theorem for MS periodic Gaussian sources. Finally, Section 6 contains some concluding remarks.

2 Preliminaries

In this section, we state the basic definitions and results in rate distortion theory and MS periodic sources which are needed for the proofs of theorems in the subsequent sections.

A. Rate Distortion Function

The n-th order rate distortion function for stationary sources is defined as follows:

Definition 1 (n-th order rate distortion function for stationary sources) *[2] Let $\mathbf{X} = [X_1, \cdots, X_n]^T$ be the n-tuple random vector consisting of r.v.'s $X_1, \cdots, X_n$ which are n consecutive elements of the stationary source $\{X_i\}$, and $P(\mathbf{x})$ be the probability density of $\mathbf{X}$. Consider all conditional probabilities $P(\mathbf{y}/\mathbf{x})$ for the reproducing words $\mathbf{y} = [y_1, \cdots, y_n]^T$ given the source words $\mathbf{x}$. The $n-th$ order rate distortion function $R_n(D)$ for $\mathbf{X}$, with respect to the single-letter distortion measure $d_n(\mathbf{x}, \mathbf{y})$, is defined as*

$$R_n(D) = \frac{1}{n} \min_{P(\mathbf{y}/\mathbf{x}) \in S} I(\mathbf{X}, \mathbf{Y}), \tag{1}$$

where

$$\begin{aligned} S &= \{P(\mathbf{y}/\mathbf{x}) : E\{d_n(\mathbf{X}, \mathbf{Y})\} \leq D\} \\ d_n(\mathbf{x}, \mathbf{y}) &= \frac{1}{n} \sum_{i=1}^{n} d(x_i, y_i) \text{ is a single-letter distortion measure,} \end{aligned}$$

and $I(\mathbf{X}, \mathbf{Y})$ is the mutual information between $\mathbf{X}$ and $\mathbf{Y}$. □

For finite-length Gaussian sources $\{X_i\}_{i=1}^n$, the n-th order rate distortion function $R_n(D)$ is also defined by eq.(1), where $\mathbf{X} = [X_1, X_2, \cdots, X_n]^T$.

Based on Definition 1, the rate distortion function of stationary sources is defined as follows:

Definition 2 *Let $\{X_i\}$ be a discrete-time stationary source. The rate distortion function $R(D)$ for $\{X_i\}$ with respect to a single-letter distortion measure is defined as*

$$R(D) = \lim_{n \to \infty} R_n(D). \tag{2}$$

where $R_n(D)$ is defined in eq.(1). □

Calculation of $R_n(D)$, in general, is very difficult. However, for finite-length independent Gaussian sources $\{X_i\}_{i=1}^n$ with variances $\{\lambda_i\}_{i=1}^n$ and the MSE distortion measure, we have [2]

$$R_n(D) = \frac{1}{n}\sum_{i=1}^{n} max(0, \frac{1}{2}\log\frac{\lambda_i}{\theta}), \tag{3}$$

where θ is specified by

$$D = \frac{1}{n}\sum_{i=1}^{n} min(\theta, \lambda_i). \tag{4}$$

The $P(\mathrm{y}/\mathrm{x})$ which achieves $R_n(D)$ is given by

$$\begin{aligned} P(\mathrm{y}/\mathrm{x}) &= \prod_{i=1}^{n} P(y_i/x_i) \qquad (5) \\ P(y_i/x_i) &= \begin{cases} \frac{1}{\sqrt{2\pi\beta_i\theta}} exp\{\frac{-(y_i-\beta_i x_i)^2}{2\beta_i\theta}\} & \text{if } \theta \le \lambda_i \\ \delta(y_i) & \text{if } \theta > \lambda_i \end{cases} \\ \beta_i &= 1 - \frac{\theta}{\lambda_i}. \end{aligned}$$

This would then imply that the pdf of the codeword is given by

$$\begin{aligned} P_y(\mathrm{y}) &= \prod_{i=1}^{n} P(y_i) \qquad (6) \\ P(y_i) &= \begin{cases} N(0, \lambda_i - \theta) & \text{if } \theta \le \lambda_i \\ \delta(y_i) & \text{if } \theta > \lambda_i \end{cases}, \end{aligned}$$

where θ can be found from eq.(4) if D is known.

Using the above results, the $R_n(D)$ of general stationary Gaussian sources can also be found analytically. For $\{X_i\}$ a zero mean stationary Gaussian sequence, $R_n(D)$, the rate distortion function of any finite portion of this sequence is equal to the rate distortion function of the independent Gaussian finite-sequence obtained by uncorrelating the finite portion of the original sequence by a unitary transformation. The variances λ_i's of the independent finite sequence are the eigenvalues of the covariance matrix of the finite portion of the original sequence. Hence, eqs.(2), (4) and (3) can be used to obtain the rate distortion function of general stationary Gaussian sources [2].

B. MS Periodic Sources

Definition 3 *[7] A source $\{X_i\}$ is called MS (mean-squared) periodic if and only if*

$$E\{|X_{k+N} - X_k|^2\} = 0 \quad for\ every\ k,$$

where N is called the period of the source $\{X_i\}$. □

The autocorrelation function of MS periodic sources has the double periodicity property given by :

Theorem 1 *[7] A process $\{X_k\}$ is MS periodic iff its autocorrelation function $\phi_{k,l}$ is such that*

$$\phi_{k+mN,l+nN} = \phi_{k,l} \tag{7}$$

for any integer m and n. □

From Theorem 1, it follows that the MS periodic sources are also cyclostationary sources, but the converse is not necessarily true.

Due to the nonstationarity of MS periodic Gaussian sources, before presenting the source coding theorem and its converse, we need to define the rate distortion function for these sources. This definition is presented in the following section.

3 Rate Distortion Function for MS Periodic Gaussian Sources

We begin this section with the definition of n-th order rate distortion function for nonstationary sources.

Definition 4 **($n-$th order rate distortion function for the nonstationary sources)** *Let* $\mathbf{X}_k = [X_k, \cdots, X_{k+n-1}]^T$ *be the n-tuple random vector consisting of r.v.'s* $X_k, \cdots, X_{k+n-1}$ *which are n consecutive elements starting at k from the nonstationary source* $\{X_i\}$*, and* $P(\mathbf{x}_k)$ *be the probability density of* $\mathbf{X}_k$*. Consider all conditional probabilities* $P(\mathbf{y}_k/\mathbf{x}_k)$ *for the reproducing words* $\mathbf{y}_k = [y_k, \cdots, y_{k+n-1}]^T$ *given the source words* $\mathbf{x}_k$*. The $n-$th order rate distortion function* $R_{n,k}(D)$ *for* $\mathbf{X}_k$*, with respect to the single-letter distortion measure* $d_n(\mathbf{x}_k, \mathbf{y}_k)$*, is defined as*

$$R_{n,k}(D) = \frac{1}{n} \min_{P(\mathbf{y}_k/\mathbf{x}_k) \in S} I(\mathbf{X}_k, \mathbf{Y}_k),$$

where

$$\begin{aligned} S &= \{P(\mathbf{y}_k/\mathbf{x}_k) : E\{d_n(\mathbf{X}_k, \mathbf{Y}_k)\} \leq D\} \\ d_n(\mathbf{x}_k, \mathbf{y}_k) &= \frac{1}{n} \sum_{i=k}^{k+n-1} d(x_i, y_i) \text{ is a single-letter distortion measure.} \end{aligned}$$

□

The n-th order rate distortion function of a nonstationary source, $R_{n,k}(D)$, can be found analytically if the source is Gaussian. Using the same transform technique used for stationary Gaussian sources, the $R_{n,k}(D)$ for $\{X_i\}$ is equal to the n-th order rate distortion function of finite-length independent Gaussian source obtained by uncorrelating $\{X_i\}_{i=k}^{k+n-1}$ through a unitary transform. Therefore, if we let $\lambda_1, \lambda_2, \cdots, \lambda_n$ be the eigenvalues of the covariance matrix of $\{X_i\}_{i=k}^{k+n-1}$, then $R_{n,k}(D)$ of the source is equal to eq.(3) given in the previous section.

Although the index k of $R_{n,k}(D)$, in general, can not be dropped for nonstationary sources, the following lemmas and theorems show that for MS periodic sources and for n a multiple of the period N, the n-th order rate distortion function is independent of k.

Lemma 1 *If* $\{X_i\}$ *is an MS periodic Gaussian source with period N, then* $R_{N,k}(D) = R_{N,k+l}(D)$ *for l any integer.*

Proof: Without loss of generality, we set $k = 0$. Let $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_l$ be the covariance matrixes of $\{X_i\}_{i=0}^{N-1}$ and $\{X_i\}_{i=l}^{l+N-1}$, respectively.

Therefore,

$$\mathbf{\Phi}_0 = \begin{bmatrix} \phi_{0,0} & \phi_{0,1} & \cdots & \phi_{0,N-1} \\ \phi_{1,0} & \phi_{1,1} & \cdots & \phi_{1,N-1} \\ \vdots & & & \vdots \\ \phi_{N-1,0} & \phi_{N-1,1} & \cdots & \phi_{N-1,N-1} \end{bmatrix}, \tag{8}$$

and

$$\mathbf{\Phi}_l = \begin{bmatrix} \phi_{l,l} & \phi_{l,l+1} & \cdots & \phi_{l,N-1} & \phi_{l,N} & \cdots & \phi_{l,l+N-1} \\ \phi_{l+1,l} & \phi_{l+1,l+1} & \cdots & \phi_{l+1,N-1} & \phi_{l+1,N} & \cdots & \phi_{l+1,l+N-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \phi_{N-1,l} & \phi_{N-1,l+1} & \cdots & \phi_{N-1,N-1} & \phi_{N-1,N} & \cdots & \phi_{N-1,l+N-1} \\ \phi_{N,l} & \phi_{N,l+1} & \cdots & \phi_{N,N-1} & \phi_{N,N} & \cdots & \phi_{N,l+N-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \phi_{l+N-1,l} & \phi_{l+N-1,l+1} & \cdots & \phi_{l+N-1,N-1} & \phi_{l+N-1,N} & \cdots & \phi_{l+N-1,l+N-1} \end{bmatrix}$$

$$= \begin{bmatrix} \phi_{l,l} & \phi_{l,l+1} & \cdots & \phi_{l,N-1} & \phi_{l,0} & \cdots & \phi_{l,l-1} \\ \phi_{l+1,l} & \phi_{l+1,l+1} & \cdots & \phi_{l+1,N-1} & \phi_{l+1,0} & \cdots & \phi_{l+1,l-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \phi_{N-1,l} & \phi_{N-1,l+1} & \cdots & \phi_{N-1,N-1} & \phi_{N-1,0} & \cdots & \phi_{N-1,l-1} \\ \phi_{0,l} & \phi_{0,l+1} & \cdots & \phi_{0,N-1} & \phi_{0,0} & \cdots & \phi_{0,l-1} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ \phi_{l-1,l} & \phi_{l-1,l+1} & \cdots & \phi_{l-1,N-1} & \phi_{l-1,0} & \cdots & \phi_{l-1,l-1} \end{bmatrix}. \tag{9}$$

The second equation in eq.(9) follows from the MS periodicity of the source. Since, Φ_l is a column and row translated version of $\mathbf{\Phi}_0$, $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_l$ have identical eigenvalues. Hence, by eq.(3), it follows that $R_{N,0}(D)$ is equal to $R_{N,l}(D)$ for any l. □

Theorem 2 *If $\{X_i\}$ is an MS periodic Gaussian source with period N, then $R_{mN,k}(D) = R_{mN,k+l}(D)$ for any $m > 0$ and l, both integers.*

Proof: Since $\{X_i\}$ is MS periodic with period N, $\{X_i\}$ is also MS periodic with period mN. Therefore, by Lemma 1, this theorem follows. □

Thus the index k in $R_{n,k}(D)$ can be dropped for MS periodic sources where n is a multiple of N. In other words,

$$R_{mN,k}(D) = R_{mN}(D) \quad \textit{for any } m > 0 \textit{ and any } k.$$

We can now define the rate distortion function $R(D)$ for MS periodic processes as follows:

Definition 5 *$R(D)$ for MS periodic Gaussian sources $\{X_i\}$ is defined as*

$$R(D) = \lim_{m \to \infty} R_{mN}(D)$$

□

To prove the uniqueness and existence of $R(D)$ for MS periodic sources, first we show the following property of $R_{mN}(D)$.

Lemma 2 *If $\{X_i\}$ is an MS Gaussian periodic sources with period N, then*

$$(m_1 + m_2)NR_{(m_1+m_2)N}(D) \leq m_1NR_{m_1N}(D) + m_2NR_{m_2N}(D) \tag{10}$$

for any m_1 and m_2, both positive integers.

Proof: This proof is similar to the proof of Theorem 9.8.1 of [1]. Let m_1 and m_2 be any positive integers. For any given $D > 0$, let $\mathbf{x}^K$ be the vector consist of K source letters, and $P_{m_1N}(\mathbf{y}^{m_1N}/\mathbf{x}^{m_1N})$ and $P_{m_2N}(\mathbf{y}^{m_2N}/\mathbf{x}^{m_2N})$ be the transition probabilities that achieve $R_{m_1N}(D)$ and $R_{m_2N}(D)$, respectively. Suppose we are transmitting $m_1N + m_2N$ letters; use P_{m_1N} for the first m_1N letters and, independently, P_{m_2N} for the last m_2N. Therefore, the corresponding test channel can be characterized by

$$\begin{aligned} &P_{(m_1+m_2)N}(\mathbf{y}^{(m_1+m_2)N}/\mathbf{x}^{(m_1+m_2)N}) = \\ &\qquad P_{m_1N}(\mathbf{y}^{m_1N}/\mathbf{x}^{m_1N})P(y_{m_1N+1}, \cdots, y_{(m_1+m_2)N}/x_{m_1N+1}, \cdots, x_{(m_1+m_2)N}). \end{aligned} \tag{11}$$

Let the source random vectors $\mathbf{X_1}$ be the first m_1N elements of the source, $\mathbf{X_2}$ be the succeeding m_2N elements, and $\mathbf{X_3}$ be the concatenation of the two. Also let the corresponding reproducing random vectors $\mathbf{Y_1}, \mathbf{Y_2}$ and $\mathbf{Y_3}$ be defined similarly.

Then, since

$$H(\mathbf{Y}_3) \leq H(\mathbf{Y}_1) + H(\mathbf{Y}_2), \text{ and}$$
$$H(\mathbf{Y}_3/\mathbf{X}_3) = H(\mathbf{Y}_2/\mathbf{X}_2) + H(\mathbf{Y}_1/\mathbf{X}_1),$$

it follows that

$$\begin{aligned} I(\mathbf{X}_3, \mathbf{Y}_3) &= H(\mathbf{Y}_3) - H(\mathbf{Y}_3/\mathbf{X}_3) \\ &\leq I(\mathbf{X}_1, \mathbf{Y}_1) + I(\mathbf{X}_2, \mathbf{Y}_2). \end{aligned}$$

By the definition of P_{m_1N} and P_{m_2N}, and the fact that $R_{n,k}(D)$ for $\{X_i\}_{i=k}^{k+n-1}$ is independent of k if n is a multiple of N, we have

$$I(\mathbf{X}_1, \mathbf{Y}_1) + I(\mathbf{X}_2, \mathbf{Y}_2) = m_1 N R_{m_1N}(D) + m_2 N R_{m_2N}(D). \tag{12}$$

Since the distortion measure

$$d_N(\mathbf{x}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - y_i)^2,$$

and $E\{d_{m_1N}(\mathbf{X}_1, \mathbf{Y}_1)\} < D$, $E\{d_{m_2N}(\mathbf{X}_2, \mathbf{Y}_2)\} < D$, it follows that

$$E\{d_{m_1N+m_2N}(\mathbf{X}_3, \mathbf{Y}_3)\} = \frac{1}{(m_1 + m_2)N} \sum_{i=1}^{(m_1+m_2)N} E\{(X_i - Y_i)^2\} \leq D. \tag{13}$$

From eqs.(12) and (13), and the fact that

$$(m_1 + m_2)N R_{m_1N+m_2N}(D) \leq I(\mathbf{X}_3, \mathbf{Y}_3),$$

we have eq.(10). □

For any sequence satisfying eq.(10), it follows that (Lemma 4.A.2 in [1]),

$$\inf_{m>0} R_{mN}(D) = \lim_{m\to\infty} R_{mD}(D).$$

Hence we have :

Theorem 3 *The rate distortion function $R(D)$ for MS periodic Gaussian sources $\{X_i\}$ is equal to*

$$R(D) = \inf_{m>0} R_{mN}(D);$$

hence it exists and is unique.

□

4 Source Coding Theorem for MS Periodic Gaussian Sources

Before stating and proving the theorem, we will present the following notation. Let $\mathbf{y}_1, \cdots, \mathbf{y}_K$ be the codewords of code B with blocklength n. Then for any source word $\mathbf{x}$, we let

$$d(\mathbf{x}/B) = \min_{\mathbf{y}\in B} d_n(\mathbf{x}, \mathbf{y})$$

be the error in coding $\mathbf{x}$ with code B. Thus the average distortion of code B, $d(B)$, is equal to

$$d(B) = E\{d(\mathbf{X}/B)\}.$$

We now consider the existence of optimal code for MS periodic Gaussian sources. First, we present the source coding theorem for finite-length independent Gaussian sources.

Theorem 4 *Given $\epsilon > 0$, $M > 0$ and $D > 0$, there exists an N such that for all finite-length independent Gaussian sources $\{X_i\}_{i=1}^{n}$ with $n > N$ and $\frac{1}{n}\sum_{i=1}^{n} \lambda_i < M$, there exists a code B with average distortion $d(B) < D + \epsilon$ and rate $R < R_n(D) + \epsilon$, where λ_i is the variance of X_i and $R_n(D)$ is the $n-th$ order rate distortion function defined for a single-letter, mean-squared error distortion measure.*

Proof: Refer to Theorem 1 in [6].

□

The following lemmas give additional properties of MS periodic sources.

Lemma 3 *For $\{X_i\}$ an MS periodic process with period N, the covariance matrix of $\{X_i\}_{i=k}^{k+N-1}$ is equal to the covariance matrix of $\{X_i\}_{i=k+lN}^{k+(l+1)N-1}$ for any k and l.*

Proof: Proof readily follows from the property of MS periodic processes given in Theorem 1.

□

Lemma 4 *For $\{X_i\}$ an MS periodic source with period N, let $\Phi_{k,mN}$ be the covariance matrix of $\{X_i\}_{i=k}^{k+mN-1}$ for any k and any $m > 0$, and $\lambda_{k,1}, \lambda_{k,2}, \cdots, \lambda_{k,mN}$ be the eigenvalues of $\Phi_{k,mN}$. Then*

$$\frac{1}{mN}\sum_{i=1}^{mN} \lambda_{k,i} = \frac{1}{N}\sum_{i=1}^{N-1} \phi_{k+i,k+i}. \tag{14}$$

Proof: From Theorem 1, it follows that

$$trace\Phi_{k,mN} = \sum_{i=0}^{mN-1} \phi_{k+i,k+i} = m\sum_{i=0}^{N-1} \phi_{k+i,k+i}.$$

Since

$$\sum_{i=1}^{mN} \lambda_i = trace\Phi_{k,mN},$$

therefore eq.(14) follows.

□

Based on Theorem 4 and the properties stated above, we have the following coding theorem for MS periodic Gaussian sources.

Theorem 5 (Source coding theorem) *Let $\{X_i\}$ be an MS periodic Gaussian source with period N and having $R(D)$ as its rate distortion function, with respect to the single-letter mean-squared error distortion measure. Then for any $\epsilon > 0$ and $D > 0$, there exists a K such that there exists a code B with blocklength kN, for $k > K$, and having rate $R < R(D) + \epsilon$ and average distortion $d(B) < D + \epsilon$.*

Proof:

Recall that in the coding process, we divide the source $\{X_i\}$ into a sequence of n-long consecutive blocks, where n is called the *blocklength*. For stationary Gaussian sources, each block has the same covariance matrix and consequently the same joint pdf. Hence only one code constructed according to the common joint pdf would be sufficient for the whole sequence. For the MS periodic Gaussian sources, although they are nonstationary, Lemma 3 implies that, if the blocklength is a multiple of the period, then each successive block also has a common pdf. Hence, as for stationary sources, only one code needs to be constructed. Furthermore, when the blocklength is a multiple of the period, by Theorem 2, it follows that the lowest rate achievable by any block of the source is identical and independent of the starting position of the block. Therefore, in the remainder of the proof, the starting position of the block is left unspecified.

Now we show the existence of the optimal code. Let $\mathbf{\Phi}_{kN}$ be the covariance matrix of any block of the source of length kN and $\lambda_i, i = 1, \cdots, kN$ be its eigenvalues. Since $\{X_i\}$ is MS periodic, it follows from Lemma 4 that $\frac{1}{kN}\sum_{i=1}^{kN}\lambda_i$ is bounded for all k. Furthermore, since $\mathbf{\Phi}_{kN}$ is symmetric, there exists a unitary matrix $\mathbf{A}$ such that $\tilde{\mathbf{X}} = \mathbf{AX}$, where $\mathbf{X} = [X_1, \cdots, X_{kN}]^T$, and $\tilde{\mathbf{X}} = [\tilde{X}_1, \cdots, \tilde{X}_{kN}]^T$, consists of kN independent r.v.'s with variances $\lambda_i, i = 1, \cdots, kN$, respectively. Now since $\frac{1}{kN}\sum_{i=1}^{kN}\lambda_i$ is bounded for k and $\{\tilde{X}_i\}_{i=1}^{kN}$ is an independent sequence, it follows from Theorem 4 that given $D > 0$ and $\epsilon/2 > 0$, there exists a K_1 such that for any $k > K_1$, there exists a code $\tilde{B}$ for $\{\tilde{X}_i\}$ with rate less than $R_{kN}(D) + \epsilon/2$ and distortion less than $D + \epsilon/2$. In addition, since the unitary transform preserves the distortion, and $R_{kN}(D)$ in both the transform domain and original domain are identical, the code B obtained by unitary transforming $\tilde{B}$ into the original domain will also have a distortion less than $D + \epsilon/2$ and rate less than $R_{kN}(D) + \epsilon/2$. Finally, since $\lim_{k\to\infty} R_{kN}(D) = R(D)$, given $D > 0$ and $\epsilon/2 > 0$, there exists a K_2 such that for all $k > K_2$, $R_{kN}(D) - R(D) < \epsilon/2$. Therefore by setting $K = \max(K_1, K_2)$, the code B has rate less than $R(D) + \epsilon$ for $k > K$. This completes the proof.

□

5 Rate Distortion Function and Converse Coding Theorem for MS Periodic Gaussian Sources

This section presents two results pertaining to MS periodic Gaussian sources. First, we prove that $R(D) = 0$ for any $D > 0$. Then, we present the converse coding theorem for these sources.

Lemma 5 *Let the eigenvalues of the covariance matrix* $\mathbf{\Phi}_1$ *of* $\{X_i\}_{i=k}^{k+N-1}$ *of an MS periodic source* $\{X_i\}$ *be* $\lambda_1, \lambda_2, \cdots, \lambda_N$*, and let the* $mN \times mN$ *matrix* $\mathbf{\Phi}_m$*,*

$$\mathbf{\Phi}_m = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \\ \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \\ \vdots & \vdots & & \vdots \\ \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \end{bmatrix},$$

be the covariance matrix of $\{X_i\}_{i=k}^{k+mN-1}$*. Then,* $m\lambda_1, m\lambda_2, \cdots, m\lambda_N, 0, \cdots, 0$ *are the eigenvalues of* $\mathbf{\Phi}_m$*.*

Proof: Let $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_N$ be the eigenvectors of $\mathbf{\Phi}_1$ associated with eigenvalues $\lambda_1, \lambda_2, \cdots, \lambda_N$, respectively. Therefore,

$$\mathbf{\Phi}_1\mathbf{v}_i = \lambda_i\mathbf{v}_i,\ i = 1, \cdots, N$$

Now, let $Nm \times 1$ vector $\mathbf{u}_i$ be

$$\mathbf{u}_i = [\mathbf{v}_i^T\ \ \mathbf{v}_i^T\ \ \cdots\ \ \mathbf{v}_i^T]^T,\ i = 1, \cdots, N.$$

Since

$$\mathbf{\Phi}_m\mathbf{u}_i = \begin{bmatrix} \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \\ \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \\ \vdots & \vdots & & \vdots \\ \mathbf{\Phi}_1 & \mathbf{\Phi}_1 & \cdots & \mathbf{\Phi}_1 \end{bmatrix} \begin{bmatrix} \mathbf{v}_i \\ \mathbf{v}_i \\ \vdots \\ \mathbf{v}_i \end{bmatrix} = \begin{bmatrix} m\mathbf{\Phi}_1\mathbf{v}_i \\ m\mathbf{\Phi}_1\mathbf{v}_i \\ \vdots \\ m\mathbf{\Phi}_1\mathbf{v}_i \end{bmatrix}$$

$$= \begin{bmatrix} m\lambda_i \mathbf{v}_i \\ m\lambda_i \mathbf{v}_i \\ \vdots \\ m\lambda_i \mathbf{v}_i \end{bmatrix} = m\lambda_i \mathbf{u}_i \ ,$$

We conclude that, $\mathbf{u}_i$, $i = 1, \cdots, N$ are the eigenvectors of $\mathbf{\Phi}_m$ and $m\lambda_i, i = 1, \cdots, N$ are the corresponding eigenvalues.

To show the remaining eigenvalues are equal to zero, we use the properties of the trace of a matrix. Since

$$trace(\mathbf{\Phi}_m) = m \ trace(\mathbf{\Phi}_1),$$

and

$$trace(\mathbf{\Phi}_1) = \sum_{i=1}^{N} \lambda_i,$$

we have

$$trace(\mathbf{\Phi}_m) = \sum_{i=1}^{N} m\lambda_i. \tag{15}$$

Because the sum of eigenvalues of a matrix is equal to its trace and $\mathbf{\Phi}_m$ is nonegative definite, it follows from eq.(15) and the fact that $m\lambda_i$, $i = 1, \cdots, N$ are N of the eigenvalues of $\mathbf{\Phi}_m$, it follows that the remaining eigenvalues are zero. □

Theorem 6 *For $\{X_i\}$ an MS periodic Gaussian source,*

$$R(D) = 0 \quad for \ any \ D > 0.$$

Proof: Recall that the n-th order rate distortion function of MS periodic Gaussian sources, for $n = N$, is equal to

$$R_N(D) \ = \ \frac{1}{N}\sum_{i=1}^{N} max(0, \frac{1}{2}\log\frac{\lambda_i}{\theta}), \text{ such that} \tag{16}$$

$$D \ = \ \frac{1}{N}\sum_{i=1}^{N} min(\theta, \lambda_i). \tag{17}$$

where we assume that $\lambda_1 \leq \lambda_2, \cdots, \leq \lambda_N$.

If the blocklength n is equal to mN, then by eq.(3) and Lemma 5, we have

$$R_{mN}(D) \ = \ \frac{1}{mN}\sum_{i=1}^{N} max(0, \frac{1}{2}\log\frac{m\lambda_i}{\phi}), \text{ such that} \tag{18}$$

$$D \ = \ \frac{1}{mN}\sum_{i=1}^{N} min(\phi, m\lambda_i)$$

$$= \ \frac{1}{N}\sum_{i=1}^{N} min(\frac{\phi}{m}, \lambda_i). \tag{19}$$

It follows from eqs.(17) and (19) that

$$\phi = m\theta. \tag{20}$$

Substituting eq.(20) into eq.(18), we have

$$\begin{aligned}
R_{mN}(D) &= \frac{1}{mN}\sum_{i=1}^{N} max(0, \frac{1}{2}\log\frac{\lambda_i}{\theta}) \\
&= \frac{1}{m}R_N(D) \\
&\rightarrow 0 \text{ as } m \rightarrow \infty.
\end{aligned}$$

□

From Theorem 6 and the fact that rate cannot be less than zero, we have the following converse source coding theorem for MS periodic Gaussian sources.

Theorem 7 *For $\{X_i\}$ an MS periodic source, the rate distortion function $R(D)$, with respect to the single-letter mean-squared error distortion measure, is zero. Therefore, for any $D > 0$ given, there exists no code B for $\{X_i\}$ with distortion $d(B) < D$ and rate $R < R(D)$.*

□

Based on the source coding theorem and its converse proved in this paper, we conclude that the $R(D)$ we defined for MS periodic Gaussian sources represents the minimum rate which can be achieved for these sources, which is consistent with usual concept of the rate distortion function $R(D)$ defined for stationary sources.

6 Conclusion

With the theorems presented in this paper, the class of Gaussian sources for which an optimal code exists has been extended to include the MS periodic (a special class of nonstationary) sources and the rate distortion function for these sources has been shown to be identically zero. In other words, the average rate of information contained in an MS periodic Gaussian source is zero.

References

[1] Gallager, R.G., *Information Theory and Reliable Communication*, Wiley, 1968.

[2] Berger, T., *Rate Distortion Theory*, Prentice Hall, 1971.

[3] Pearlman, W.A. and Jakatdar, P., "A transform tree code for stationary Gaussian sources," *IEEE Trans. Inform. Theory*, vol. IT-31, pp.761-768, Nov. 1985.

[4] Mazor, B. and Pearlman, W.A., "A trellis code construction and coding theorem for stationary Gaussian sources," *IEEE Trans. Inform. Theory*, vol. IT-29, pp.924-930, Nov. 1983.

[5] Mazor, B. and Pearlman, W.A., "A tree coding theorem for stationary Gaussian sources and the squared-error distortion measure," *IEEE Trans. Inform. Theory*, vol. IT-32, pp.156-165, March 1986.

[6] Hwang, W.-J. and Derin, H., "Some coding theorems for Gaussian sources," *Proc. 25th Annu. Conf. Inf. Sci. Syst.*, Princeton, N.J., March 1992.

[7] Papoulis, A.,*Probability, Random Variables, and Stochastic Processes*, 3rd Edition, McGraw-Hill, 1989.

Experiments in Hyper-Redundant Manipulation

Gregory S. Chirikjian[†] Joel W. Burdick

School of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125

Abstract

"Hyper-redundant" robots have a very large degree of kinematic redundancy. These robots are analogous to "snakes," "elephant trunks," and "tentacles." Potential benefits of hyper-redundancy include superior obstacle avoidance ability, novel forms of locomotion, and robust tentacle-like grasping. In previous work, the authors have developed a novel and efficient framework for hyper-redundant robot kinematic analysis and algorithm development. This paper reviews the implementation of 30 degree-of-freedom planar hyper-redundant robot which demonstrates and validates this earlier work. Experiments in manipulation, obstacle avoidance, locomotion, grasping, and grasped object reorientation are summarized.

1. Introduction

Hyper-redundant robots have previously been called 'highly articulated,' 'tentacle,' 'snake-like,' 'tensor-arm,' 'elephant-like,' and 'spine.' The authors have coined the word 'hyper-redundant' to capture the essence of these related concepts. To date, hyper-redundant robots have remained largely a laboratory curiosity because: (1) previous kinematic modeling techniques have not been efficient or well suited to hyper-redundant robot task modeling; (2) the mechanical design and implementation of hyper-redundant robots has been perceived as unnecessarily complex; and (3) hyper-redundant robots are not anthropomorphic and thus require different programming paradigms. The authors have undertaken a broadly based investigation which we hope will eventually overcome many of the obstacles to practical hyper-redundant robot deployment. In previous papers, we have presented novel techniques for unified kinematic analysis of hyper-redundant robots [1-5]. These techniques have also been used as the basis of novel obstacle avoidance [2], locomotion [4], and grasping schemes [4] (these papers should be consulted for references to other approaches to hyper-redundant mechanism analysis and previous developments of hyper-redundant robot prototypes).

This paper briefly reviews our previous work and presents the design and implementation of a thirty degree-of-freedom (DOF) planar hyper-redundant robot which was developed to verify and validate our analytical approach. We also present experiments in hyper-redundant manipulation, obstacle avoidance, locomotion, grasping, object manipulation, and dual arm hyper-redundant manipulation. Some of these operating modes, such as grasped object reorientation, have not previously been considered or demonstrated in the literature.

2. Review of Planar Hyper-Redundant Robot Kinematics

[†] Currently at the Dept. of Mechanical Engineering, 124 Latrobe Hall, Johns Hopkins University, Baltimore, MD 21218.

Our kinematic analysis method is based on a *backbone curve* which captures the hyper-redundant robot's important macroscopic features. Kinematic analysis and task planning operations reduce to the determination of the proper time varying behavior of the backbone curve. Continuous backbone curves shapes which result from planning algorithms can be used to directly control the geometry of continuous morphology robots. For discretely segmented morphologies, such as the variable geometry truss (VGT) robot described in this paper, the backbone curve nominally captures the robot geometry closely, though not exactly. *Fitting algorithms* (Section 2.3) are used to determine the discrete actuator displacements which will cause the robot to adhere as closely as possible to the continuous backbone curve solution. In this paper, we focus on planar hyper-redundant kinematics, as our actual robot is planar.

2.1. Parametrization of the Backbone Curve

Hyper-redundant robot backbone curves, can be *nonextensible* (fixed length), or *extensible* (variable length). An intrinsic backbone curve parametrization consists of both the parametrization of points along the curve and the choice of a reference frame at each curve point. Points on both extensible and nonextensible planar and spatial curves can be parametrized in the form:

$$\overline{x}(s,t) = \int_0^s l(\sigma,t)\overline{u}(\sigma,t)d\sigma \tag{2.1}$$

where $s \in [0,1]$ measures distance along the backbone curve. $\overline{x}(s,t)$ is a vector from the robot base to the point s on the backbone curve. $\overline{u}(s,t)$ is the unit vector tangent to the curve at s. $l(s,t)$ is the length of the curve tangent and takes the form:

$$l(s,t) = 1 + \epsilon(s,t) > 0 \tag{2.2}$$

where $\epsilon(s,t)$ is the local extensibility of the robot at s. $\epsilon(s,t) > 0$ indicates local extension, while $\epsilon(s,t) < 0$ indicates contraction. The parametrization of Equation (2.2) has the following interpretation. The backbone curve is "grown" from the base by propagating the curve forward along the tangent vector, which is varying its direction according to $\overline{u}(s,t)$ and varying its magnitude (or 'growth-rate') according to $l(s,t)$.

In the planar case, let $\overline{x}(s,t) = [x(s,t), y(s,t)]^T$ denote the position of point s with respect to the backbone curve base. Thus:

$$x(s,t) = \int_0^s l(\sigma,t)\sin\theta(\sigma,t)d\sigma; \qquad y(s,t) = \int_0^s l(\sigma,t)\cos\theta(\sigma,t)d\sigma. \tag{2.3}$$

$\theta(s,t)$ is the clockwise measured angle which the tangent to the curve at s makes with the y-axis. It takes the form:

$$\theta(s,t) = \int_0^s \kappa(\sigma,t)l(\sigma,t)d\sigma \tag{2.4}$$

where $\kappa(s,t)$ is the classical *curvature function* of the backbone curve. $l(s,t)$ and $\theta(s,t)$ (or alternatively, $l(s,t)$ and $\kappa(s,t)$) are termed the *intrinsic shape functions* of the backbone curve, as they completely describe the backbone curve shape.

A *backbone reference frame* at s is a set of right-handed orthonormal vectors with frame origin coincident with point s in the curve. In the planar case, the reference frame basis vectors are the unit backbone curve tangent and normal vectors:

$$\mathbf{Q}(s,t) = \begin{bmatrix} \sin\theta(s,t) & \cos\theta(s,t) \\ \cos\theta(s,t) & -\sin\theta(s,t) \end{bmatrix} \tag{2.5}$$

2.2. Inverse Kinematic Schemes

For given shape functions, the forward kinematics can be found by integration of Equation (2.1). However, the inverse kinematic problem, which corresponds to finding a backbone curve which satisfies tasks constraints, is more difficult. There are an infinite number of such curves. Previous work has focused on two methods for resolving hyper-redundancy: a "modal" approach, and one based on the calculus of variations.

In the modal approach, the shape functions are restricted to the 'modal' form:

$$\theta(s,t) = \sum_{i=1}^{N_\theta} a_i(t)\Phi_i(s) \qquad l(s,t) = \sum_{i=N_\theta+1}^{N} a_i(t)\Phi_i(s) \tag{2.6}$$

where Φ_i is a *mode function*, and a_i is a *modal participation factor*. N equals or exceeds the total number of task constraints. Hyper-redundancy is resolved in the modal approach by constraining the backbone curve to N effective DOF. In the planar case, these DOF are distributed between N_θ 'bending modes' and $(N - N_\theta)$ 'extension' modes. The $\{\Phi_i\}$ are predetermined functions chosen by the programmer. Thus, the backbone curve geometry becomes solely a function of the $\{a_i\}$. The inverse kinematics problem reduces to finding the $\{a_i\}$ which satisfy task constraints.

The restriction to modal form is practically useful, as it often leads to closed or nearly closed form inverse kinematic solutions. For example, the following choice of modes for $N_\theta = 2$ (a nonextensible planar backbone curve where the end-effector position is of concern only):

$$\Phi_1(s) = \sin 2\pi s; \qquad \Phi_2(s) = 1 - \cos 2\pi s \tag{2.7}$$

Substituting these two modes into Equations (2.6) and (2.3) and evaluating at $s = 1$, it can be shown that the "forward kinematic" equations reduce to:

$$\begin{aligned} x_{ee} &= x_1(1) = \sin(a_2) J_0\left[(a_1^2 + a_2^2)^{\frac{1}{2}}\right] \\ y_{ee} &= x_2(1) = \cos(a_2) J_0\left[(a_1^2 + a_2^2)^{\frac{1}{2}}\right] \end{aligned} \tag{2.8}$$

where J_0 is the zeroth order Bessel function. The 'inverse kinematics' (evaluation of modal participation factors) can be computed as:

$$a_1 = \pm\left(\left[J_0^{-1}\left[(x_{ee}^2 + y_{ee}^2)^{\frac{1}{2}}\right]\right]^2 - \left[\mathrm{Atan2}\,(x_{ee}, y_{ee})\right]^2\right)^{\frac{1}{2}} \tag{2.9}$$

$$a_2 = \mathrm{Atan2}\,(x_{ee}, y_{ee}). \tag{2.10}$$

J_0^{-1} is the 'restricted inverse Bessel function of zero order', and is defined as the inverse of $J_0(x)$ for $0 < x < \mu$ where $\mu \approx 3.832$ is the first local minimum of J_0. The plus and minus sign distinguishes two possible poses. Other poses would exist if other intervals of the argument were permitted in computing the inverse of $J_0(x)$. Other examples of closed form inverse kinematics can be found in [1,3,4]. It is not necessary to have closed form inverse kinematic solutions to use the modal method. Look-up tables and interpolation can store the mapping between participation factors and end-effector coordinates for any given set of modes. Alternatively, a numerical inverse kinematic solution method based on a *modal Jacobian* and a scheme analogous to resolved rate trajectory planning can be used [1,3].

Alternatively, the calculus of variations can be used to generate necessary conditions for the optimal backbone curve shape with respect to task constraints and a user-defined optimality criterion [5]. Reference [5] considers optimality criteria based on a weighted measure of backbone bending. In the planar case, the minimization of the integral:

$$I = \frac{1}{2}\int_0^1 \alpha(s)\kappa^2(s)ds = \frac{1}{2}\int_0^1 \alpha(s)\dot{\theta}^2(s)ds \tag{2.11}$$

will minize the bending of a nonextensible backbone curve. $\alpha(s)$ is a scalar weighting function, which may be chosen to reflect practical constraints [5]. In this approach, we seek to find the intrinsic function $\theta(s)$ which minimizes (2.11) while satisfying the end-effector positioning constraints of Equation (2.3) and the boundary conditions $\theta(s=0)=0$ and $\theta(s=1)=\theta_{ee}$, where θ_{ee} is the desired orientation of the backbone curve tip, or end-effector. [5] should be consulted for the solution to his problem and related examples.

2.3. 2.3. Backbone Curve "Fitting" Procedures

For a VGT morphology (and other discretely segmented mechanisms), a "fitting" procedure is used to determine the individual actuator displacements which cause the VGT to adhere as closely as possible to the continuous backbone curve solution. Figure 1 shows one module of a planar VGT robot which consists of a serial chain of n such modules. Attached to each module is an intrinsic 'input' and 'output' frame. Fitting is done by constraining the module input/output frames to coincide exactly with the intrinsic frames of the backbone curve at $(n+1)$ points, $\{s_i\}$ $i = 0, \cdots, n$. Without loss of generality, these points are chosen uniformly, e.g. $s_i = i/n$. [5] outlines a procedure for optimally reparametrizing the length distribution function so that the physical locating of the fitting points may be nonuniform.

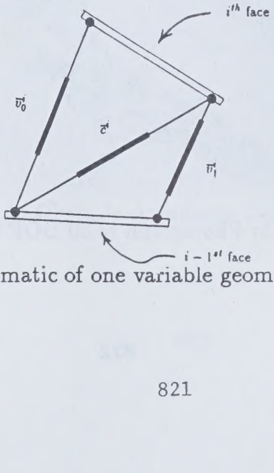


Figure 1: Schematic of one variable geometry truss module

The intrinsic frames at s_{i-1} and s_i define an inverse kinematic problem for the i^{th} truss module. This module inverse kinematic problem can be solved as follows. Let $\vec{r}^{\,i}_{i-1}$ be a vector from $\overline{x}(s_{i-1},t)$ to $\overline{x}(s_i,t)$:

$$\vec{r}^{\,i}_{i-1}(t) = \mathbf{Q}^T(s_{i-1},t)[\overline{x}(s_i,t) - \overline{x}(s_{i-1},t)] \tag{2.12}$$

The three vectors collinear with the three prismatic joint axes (Figure 1) can be computed as:

$$\begin{aligned} \vec{v}^{\,i}_j &= \vec{r}^{\,i}_{i-1} - \vec{n}^{\,i}_j + \mathrm{ROT}(-\overline{e}_3, \delta\theta^i)\vec{n}^{\,i}_j \qquad j = 1,2 \\ \vec{c}^{\,i} &= \vec{r}^{\,i}_{i-1} - \vec{n}^{\,i}_1 + \mathrm{ROT}(-\overline{e}_3, \delta\theta^i)\vec{n}^{\,i}_2 \end{aligned} \tag{2.13}$$

where $\delta\theta^i = \theta(s_i,t) - \theta(s_{i-1},t)$ and $\vec{n}^{\,i}_j$ are the vectors to the j^{th} vertex of the i^{th} platform in the frame affixed to that platform. For this example, $\vec{n}^{\,i}_1 = [-w_0/2, 0]^T$ and $\vec{n}^{\,i}_2 = [w_0/2, 0]^T$ where w_0 is the width of each horizontal face of the truss. The controlled degrees of freedom are the lengths of the prismatic actuators:

$$\lambda^i_j = ||\vec{v}^{\,i}_j||; \qquad \lambda^i_3 = ||\vec{c}^{\,i}|| \tag{2.14}$$

for $i = 1, ..., n$, and $j = 1, 2$. Thus, Equations (2.13) and refactuators provide the inverse kinematics solution for this module geometry based on the backbone curve geometry.

3. Manipulator Description

A design study of possible morphologies which would enable us to demonstrate the algorithms developed in our previous work concluded that a variable geometry truss morphology would meet all of our requirements. Our final design, Figure 2, is a 30 DOF VGT structure consisting of 10 identical modules which are planar 3 DOF parallel manipulators (with the same kinematic structure of Figure 1). The prismatic actuators are implemented with D.C. servo motors and lead screw drives. The actuators can vary in length from a minimum of 12 inches to a maximum of 18 inches. The geometry of the module can be simply modified to trade off between module dexterity and mechanical advantage.

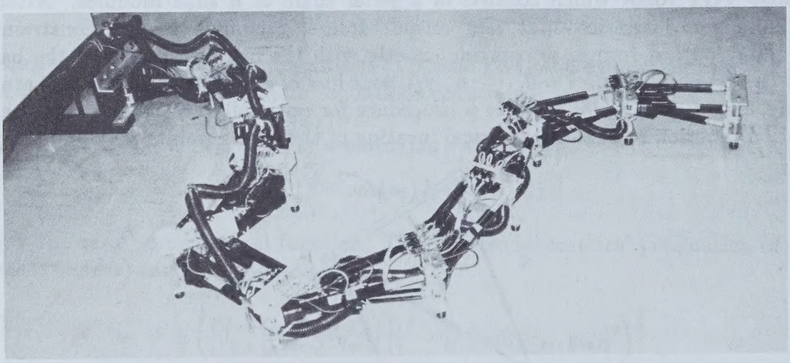


Figure 2: Photograph of 30 DOF Robot

For manipulation and grasping experiments, castors are attached to the fixed elements of the modules in order to allow low friction motion over the laboratory floor, which is the plane of manipulation. For locomotion experiments, the manipulator is detached from its base, and turned on its side. Each module is equipped with rubber faced ratchet wheels which permit rolling in one direction only. This is required for standing wave locomotion [4]. For traveling wave locomotion, the wheels are locked in place, and the wheels' rubber surfaces are use as passive feet. These rubber surfaces are also used as the contact points for planar grasping operations. Because of its modularity, the robot can be separated into two 15 DOF robots for cooperative manipulation experiments [1].

The robot system is controlled by an a multiprocessing computer, which consists of Heurikon 68030 processing boards in a VME bus, which is in turn connected to a Sun Microsystems 4/260 computer. The WindRiver Systems VxWorks software system provides operating system and system management functions. The 68030 processors compute the kinematic, path planning, sensor processing, and task level coordination functions. They also implement the low level actuator position feedback control. The Sun computer is used primarily for user interface and data storage.

4. Experiments

It is difficult to summarize these experiments in this format, and interested readers can obtain copies of a video-tape of these experiments from the authors. Figure 3 shows one snap-shot of an obstacle avoidance maneuver which employs the the theory developed in [2]. Figure 4 shows a snap-shot of the robot displaying locomotion based on the standing wave idea developed in [4,7]. The photo in Figure 5 shows how a hyper-redundant manipulator can be used as a grasping device, which can be used to maintain a 'power grasp' while still being able to incrementally rotate the object within the grasp, as formulated in [1,4,8].

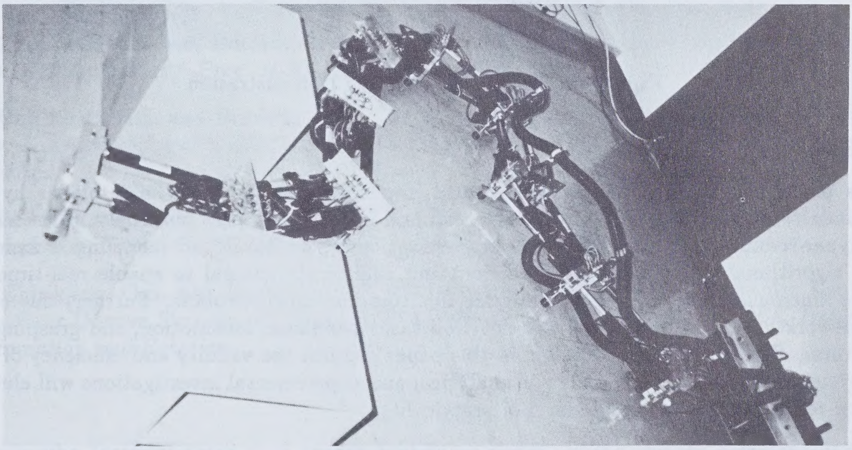


Figure 3: Obstacle Avoidance Demonstration

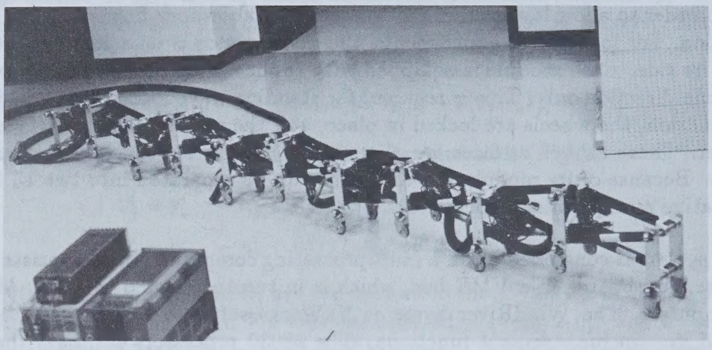


Figure 4: Standing Wave Locomotion Demonstration

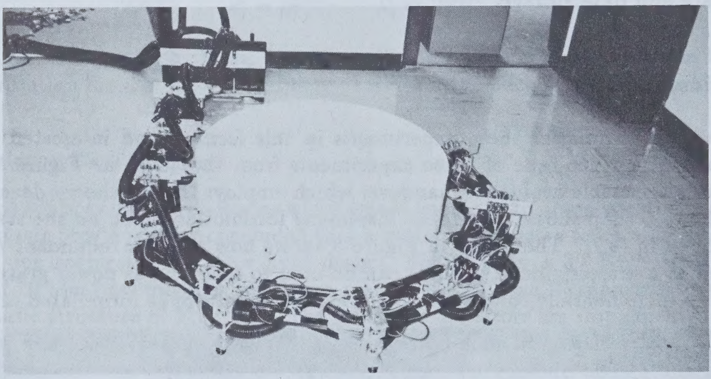


Figure 5: Tentacle-Like Grasp Demonstration

5. Summary

This paper summarized the kinematics and implementation of a 30 DOF planar hyper-redundant robot which was developed to validate the authors' previous analytical research in hyper-redundant robotic systems. In previous work, we developed modeling techniques and algorithms which were highly efficient and sufficiently general to enable real-time inverse kinematics and motion planning for this complex class of robots. Further, this basic framework has been used to develop novel obstacle avoidance, locomotion, and grasping algorithms. The experiments detailed in this paper confirm the validity and efficiency of our previous work. Our hope is that our analytical and experimental investigations will elevate hyper-redundancy to a greater level of practicality.

As demonstrated by our design, hyper-redundant robots have some inherent advantages. First, they are amenable to modular implementation, and thus not unnecessary complex to design or build. Second, only a small fraction of the robot actuators are typically in use, minimizing their duty cycle. Such systems are highly fault tolerant with respect to numerous actuator failures. For instance, ninety percent of the actuators can fail in our robot, yet the system will retain the ability to position and orient the end-effector (though obstacle avoidance capability and workspace volume will be severely reduced). Third, assembly and maintenance of this hyper-redundant manipulator is extremely simple. The entire mechanism can be assembled from its constituent parts in less than one half hour. Actuator replacement requires less than 5 minutes. Thus, repairability of such robots is advantageous for field deployment. Such systems can also be weight efficient. This robot can extend to a maximum length of 18 feet, and yet it weighs less than 55 kg.

Aknowledgements: This work was sponsored in part by a NASA Graduate Student Researchers Program fellowship (for the first author), NSF grants MSS-901779 and MSS-9157843, and an office of Naval Research Young Investigator Award N00014-92-J-1920.

6. References

[1] Chirikjian, G.S., “Theory and Applications of Hyper-Redundant Robotic Manipulators,” Ph.D Thesis, Dept. of Applied Mechanics, California Institute of Technology, June 1992.

[2] Chirikjian, G.S., Burdick, J.W., “An Obstacle Avoidance Algorithm for Hyper-Redundant Manipulators,” *Proc. IEEE Int. Conf. on Robotics and Automation* (ICRA), Cincinnati, OH, May 13-18, 1990.

[3] Chirikjian, G.S., Burdick, J.W., “ Parallel Formulation of the Inverse Kinematics of Modular Hyper-Redundant Manipulators,” *Proc. ICRA*, Sacramento, CA, April, 1991.

[4] Chirikjian, G.S., Burdick, J.W., “Kinematics of Hyper-Redundant Locomotion with Applications to Grasping,” *Proc. ICRA*, Sacramento, CA, April, 1991.

[5] Chirikjian, G.S. and Burdick, J.W., “Kinematically Optimal Hyper-Redundant Manipulator Configurations,” *Proc. ICRA*, Nice, France, May, 1992.

[6] Chirikjian, G.S. and Burdick, J.W., “A Modal Approach to Hyper-Redundant Manipulator Kinematics,” (with G.S. Chirikjian), submitted to *IEEE Trans. on Automation and Robotics.*

[7] Chirikjian, G.S. and Burdick, J.W., “The Kinematics of Planar Hyper-Redundant Robotic Locomotion,” submitted to *IEEE Trans. on Automation and Robotics.*

[8] Chirikjian, G.S. and Burdick, J.W., “Curvilinear Hyper-Redundant Robotic Locomotion Over Uneven Terrain, with Applications to Grasping,” submitted to *IEEE Trans. on Automation and Robotics.*

Kinematic Design of Fault Tolerant Manipulators

Wing K.F. Au* Christiaan J.J. Paredis* Pradeep K. Khosla†

Department of Electrical and Computer Engineering and
The Robotics Institute,
Carnegie Mellon University,
Pittsburgh, PA 15213

Abstract: *We define the property of fault tolerance and develop a general constructive proof of the existence of fault tolerant manipulators. After formulating the necessary and sufficient condition for fault tolerance, we design a planar manipulator with a minimal kinematic structure. Finally, we show how our approach can be generalized and we develop a numerical method for designing spatial fault tolerant manipulators.*

1 Introduction

Consider the use of manipulators in a nuclear environment where equipment has to be repaired or space has to be searched for radioactive contamination. A manipulator that is deployed for this kind of critical missions must have many desirable attributes. It must be *versatile*. That is, the manipulator system must be capable of accomplishing a wide variety of tasks and possibly be capable of reconfiguring itself. It must be *easily maintainable* and cost effective. Ease of maintenance would significantly reduce the downtime and therefore increase the utility of the system. It must also be *reliable or fault tolerant*. That is, if one of the subsystems were to fail, the reduced system should still be capable of accomplishing its mission objectives. The property of fault tolerance is a must in critical missions that may involve high costs if the mission were to fail due to a failure in the manipulator system [14].

In this paper, we propose a definition of fault tolerance and show how any manipulator can be made fault tolerant. Since we are particularly interested in minimal kinematic structures, we prove that a 5-degree-of-freedom (DOF) manipulator is sufficient to reach any planar set of positions/orientations in a fault tolerant way. Finally, for the more complex case of spatial manipulator design, a numerical approach is developed and illustrated with two examples.

2 Definition of Fault Tolerance

In order to set the stage for our development, we define the following properties of fault tolerant manipulators [1]:

- **Fault Tolerant Manipulator**: An n-DOF manipulator that will still be able to meet the task specifications, even if any one or more of its joints fail and are frozen at any arbitrary joint angles.
- **Reduced Order Derivative (ROD)**: When some joints of a manipulator fail, the effective number of joints is smaller than the initial number of joints. This faulty manipulator is called a reduced order derivative.
- **Order of Fault Tolerance**: An n-DOF manipulator is fault tolerant of the i-th order, if and only if all $(n-i)$ DOF reduced order derivatives can still perform the specified task.
- **Fault Tolerant Work Space**: The fault tolerant work space of an i-th order fault tolerant n-DOF manipulator is the intersection of the work spaces of all possible $(n-i)$ DOF reduced order derivatives.

These definitions differ from the concept of fault tolerance as proposed by Maciejewski [11]. Instead of attributing the property of fault tolerance to a *manipulator*, he quantifies a measure of fault tolerance for a manipulator *posture* and describes a technique to determine the optimal fault tolerant posture, based on the singular value decomposition of the Jacobian matrix. If a joint fails in this optimal posture, the resulting reduced order derivative will have maximum possible dexterity. However, a failure at a different angle may make the execution of the task impossible.

---

*Department of Electrical and Computer Engineering and The Robotics Institute.

†Associate Professor, Department of Electrical and Computer Engineering and The Robotics Institute.

3 Existence of Fault Tolerant Manipulators

In this section, we give a constructive proof of the existence of fault tolerant manipulators. The resulting sufficient condition for k-th order fault tolerance is not a necessary condition. Therefore, it is possible that for a certain task, a fault tolerant manipulator with fewer degrees-of-freedom exists. Finding an analytical expression for the solution with the *fewest* degrees-of-freedom becomes intractable in the general case, but can be achieved for first order fault tolerance in the planar case.

3.1 In General

An obvious way to achieve fault tolerance is to design every joint with one or more redundant actuators [14]. If one of the actuators in any joint were to fail, the redundant actuator would take over and the manipulator will still be functional. This can be proven rigorously.

Suppose that we have an n-DOF manipulator, $\mathcal{M}_n$, defined by the Denavit-Hartenberg (D-H) parameters (d_i, a_i and α_i are the offset, link length and twist angle, respectively, of the i-th DOF, listed in the i-th column),

$$DH_n = \begin{pmatrix} d_1 & d_2 & \dots & d_n \\ a_1 & a_2 & \dots & a_n \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \end{pmatrix} \tag{1}$$

such that it satisfies all the kinematic requirements of a given task. It has been shown that such a manipulator can be found by using the task based design approach described in [12] or [8]. We can now construct a $2n$-DOF first order fault tolerant manipulator, $\mathcal{M}_{2n}$, with D-H parameters

$$DH_{2n} = \begin{pmatrix} 0 & d_1 & 0 & d_2 & \dots & 0 & d_n \\ 0 & a_1 & 0 & a_2 & \dots & 0 & a_n \\ 0 & \alpha_1 & 0 & \alpha_2 & \dots & 0 & \alpha_n \end{pmatrix} \tag{2}$$

as illustrated in Figure 1.

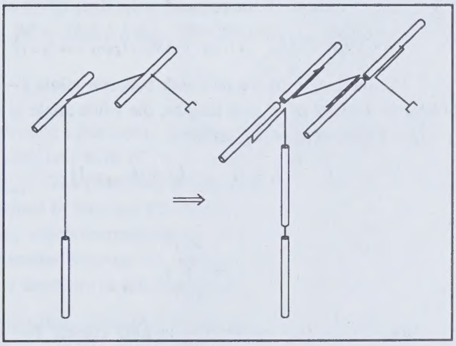


Figure 1: An example of forming a six DOF first order fault tolerant manipulator from a three DOF manipulator.

Any posture of $\mathcal{M}_n$, defined by the set of joint angles, $(\theta_1, \theta_2, \dots, \theta_n)^T$, can also be assumed by $\mathcal{M}_{2n}$, with joint angles $(\theta_1, 0, \theta_2, \dots, 0, \theta_n, 0)^T$. Even if the i-th joint fails at an arbitrary angle, θ_i^f, where a superscript f indicates an angle at which a joint is frozen, all the kinematic requirements remain satisfied for $\mathcal{M}_{2n}$ in the posture defined by the joint angles

$$
\begin{aligned}
&(\theta_1, 0, \ldots, \theta_i^f, (\theta_p - \theta_i^f), \ldots, 0, \theta_n, 0)^T \quad \text{if } (i = 2p - 1) \text{ is odd} \\
&(\theta_1, 0, \ldots, (\theta_p - \theta_i^f), \theta_i^f, \ldots, 0, \theta_n, 0)^T \quad \text{if } (i = 2p) \text{ is even}
\end{aligned}
\tag{3}
$$

This can be proven by showing the equality of the link transformation matrices of $\mathcal{M}_n$ and $\mathcal{M}_{2n}$. Similar to the construction of a first order fault tolerant manipulator, a k-th order fault tolerant manipulator can be constructed by duplicating every degree of freedom k times, which results in a $(k+1)\,n$-degree-of-freedom manipulator.

3.2 Planar Case

With the proof of existence of fault tolerant manipulators in hand, we now solve for an analytical solution for a planar first order fault tolerant manipulator with a minimum number of degrees-of-freedom. We are interested in a minimal structure because of simplicity in building and controlling the robot.

Consider a task defined by a set of positions/orientations, $W = \{(x_j, y_j, \varphi_j)\}$, that have to be reachable by the n-DOF manipulator, $\mathcal{M} = (l_1, l_2, \ldots, l_n)$, which is required to be first order fault tolerant. Before trying to solve the problem: "Find $\mathcal{M}$, given W" (=design), we develop a set of necessary and sufficient conditions for first order fault tolerance: "Given W, is $\mathcal{M}$ fault tolerant?" (=analysis). A first step in the mathematical construction of these conditions is to split the manipulator into two parts, namely, the last link, l_n, and the first $(n-1)$ links, which can be considered as a new manipulator, $\mathcal{M}^* = (l_1, l_2, \ldots, l_{n-1})$. This manipulator has to be able to reach the points in the new workspace, W^*, in *any* orientation when all its joints are operative, and in *at least one* orientation when any one of its $(n-1)$ joints is frozen at an arbitrary angle. W^* is defined as

$$
W^* = \{(R_j, 0) \mid R_j = \sqrt{(x_j - l_n \cos\varphi_j)^2 + (y_j - l_n \sin\varphi_j)^2}, \forall (x_j, y_j, \varphi_j) \in W\}
\tag{4}
$$

This equivalent positional problem is easier to solve due to radial symmetry, and can be expressed mathematically as

$$
R^C \overset{?}{\le} R_j \overset{?}{\le} R^F \quad \forall j
\tag{5}
$$

where R^C and R^F are respectively the closest and farthest radii reachable by $\mathcal{M}^*$ in a first order fault tolerant way. Thus, the analysis problem is solved by checking whether all the radii R_j in W^* are in the interval $[R^C, R^F]$.

To find an expression for R^F, we have to look at all the reduced order derivatives of $\mathcal{M}^*$:

$$
\mathcal{M}_i(\theta_i^f) = (l_1, \ldots, l_{i-2}, \mathcal{L}_i(\theta_i^f), l_{i+1}, \ldots, l_{n-1})
\tag{6}
$$

with $|l_{i-1} - l_i| \le \mathcal{L}_i(\theta_i^f) \le (l_{i-1} + l_i)$, the length of the new link between joints $i-1$ and $i+1$. Since the maximum reach of a planar manipulator is the sum of its link lengths, the worst angle at which joint i can be frozen is $\theta_i = \pi$, for which $\mathcal{L}_i = |l_{i-1} - l_i|$. The corresponding radius is

$$
R_i^F = L - (l_{i-1} + l_i) + |l_{i-1} - l_i|
\tag{7}
$$

with

$$
L = \sum_{k=1}^{n-1} l_k
\tag{8}
$$

the total length of the manipulator, $\mathcal{M}^*$. This expression is also correct for $i = 1$ and $i = n$, if we set $l_0 = l_n \to \infty$. The final expression for R^F is equal to the farthest reach, R_i^F, of the worst reduced order derivative:

$$
R^F = \min_{i = 1, n} R_i^F
\tag{9}
$$

The expression for R^C is based on the fact that the minimum distance between the 2 endpoints of a planar chain of rigid links is:

$$\max\ \{0, \text{length of the longest link} - \sum \text{lengths of the other links}\} \tag{10}$$

Since a planar manipulator is similar to a chain, we can use this fact to determine R^C as the minimum distance between the 2 endpoints of the manipulator, $\mathcal{M}^*$, namely, the base and the wrist.

We have to consider 2 possible angles at which to freeze joint i:

1. if $\mathcal{L}_i$ is the longest link, the worst angle is $\theta_i = 0$, for which $\mathcal{L}_i = l_{i-1} + l_i$.
2. if $\mathcal{L}_i$ is not the longest link, the worst angle is $\theta_i = \pi$, for which $\mathcal{L}_i = |l_{i-1} - l_i|$.

This results in a radius

$$R_i^C = \max\ \{0, 2(l_{i-1} + l_i) - L, 2l_{\max} - L + (l_{i-1} + l_i) - |l_{i-1} - l_i|\} \tag{11}$$

where $l_{\max}$ is the length of the longest link. This expression is also correct for $i = 1$ and $i = n$, if we set $l_0 = l_n = 0$.

The final expression for R^C is:

$$R^C = \max_{i = 1, n} R_i^C \tag{12}$$

Now that we have analytical expressions for R^C and R^F, we can express the necessary and sufficient condition for first order fault tolerance of a manipulator, $\mathcal{M} = (l_1, \ldots, l_n)$, with respect to a task W, as

$$R^C \le R_j \le R^F \qquad \forall R_j \in W^* \tag{13}$$

where R^C, R^F and W^* are defined by the expressions (12), (9) and (4), respectively.

In order to solve the design problem, we first have to determine the minimum number of degrees-of-freedom. Consider a 4-DOF manipulator. Equations (9) and (12), for the 3-DOF positional structure simplify to:

$$R^C = \max\ \{0, l_1 + l_2 - l_3, l_1 - |l_2 - l_3|, l_3 + l_2 - l_1, l_3 - |l_2 - l_1|\} \tag{14}$$

$$R^F = \min\ \{l_1 + l_2 - l_3, l_1 + |l_2 - l_3|, l_3 + l_2 - l_1, l_3 + |l_2 - l_1|\} \tag{15}$$

As a result, the interval $[R^C, R^F]$ reduces to the empty set, except for manipulators parametrized by $\mathcal{M}^* = (t, R, t)$ with $t \in [0, \infty)$, for which $[R^C, R^F] = \{R\}$. This means that a 4-DOF fault tolerant manipulator only exists when a hand length, l_4, can be found such that the desired radial distance of the wrist is equal for all the task points in W. This is a very particular case, so that in general, a 5-DOF manipulator is needed. A simple design can be parametrized by $\mathcal{M} = (l, l, l, l, l_5)$. The last link, l_5, should be chosen such that

$$R_{\max} = \max_j R_j \tag{16}$$

is minimal. Furthermore, by selecting a positional structure consisting of four links of equal length, we obtain a fault tolerant workspace without holes, i.e., with $R^C = 0$. Finally, the length, l, of the links should be larger than $R_{\max}/2$, so that $R^F = 2l \ge R_{\max}$. The positional structure of this design is illustrated in Figure 2. The radius of the workspace of the ROD obtained by freezing the second joint angle at $\theta_2 = \pi$ is $R^F = 2l$. Both ROD's are also able to reach back to the origin, which corresponds to the property that $R^C = 0$. When the second joint angle is fixed at zero, the workspace diameter becomes $4l$. However, the fault tolerant workspace is the intersection of the workspaces of all ROD's and is therefore in this case equal to the region inside the circle with radius $R^F = 2l$.

Finally we would like to point out that, although a 4-DOF planar manipulator is redundant, in general it will not be fault tolerant, as we proved above. On the other hand, a fault tolerant manipulator is necessarily always redundant. This means that the set of manipulators that are fault tolerant for a specific task is always a subset of the set of manipulators that are redundant for the same task.

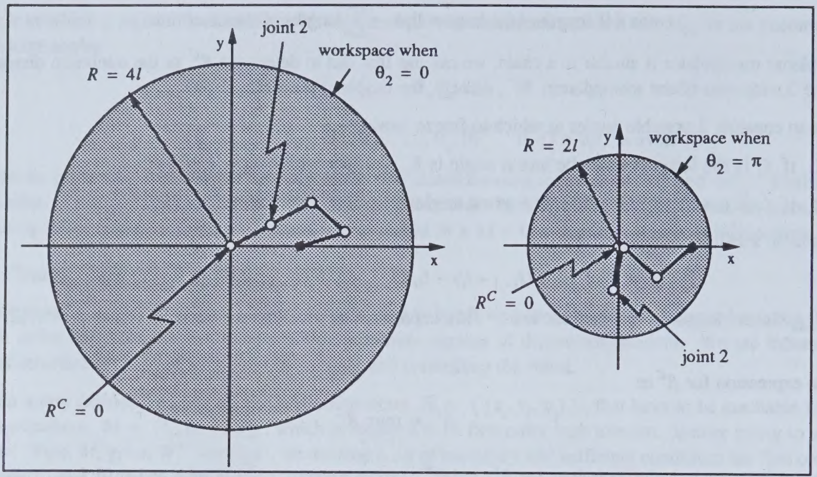


Figure 2: Example of the workspaces of two ROD's, obtained by fixing $\theta_2 = \pi$ or $\theta_2 = 0$.

4 Task Based Design of Fault Tolerant Manipulators

4.1 Mathematical Formulation

In the previous section, we were able to find necessary and sufficient conditions for fault tolerance in the planar case. In this section, we attack the spatial case, namely, we formulate a method to determine the kinematic structure of a manipulator which is fault tolerant with respect to the task of reaching a set of positions/orientations, $W = \{\mathbf{p}_j = (x_j, y_j, z_j, \vartheta_j, \varphi_j, \psi_j)\}$. Since the geometric complexity increases drastically for spatial manipulators, an analytical solution becomes infeasible. Therefore, we must choose a numerical approach to solve the design problem.

We present a technique based on the design method, proposed in [13]. In this approach, the Denavit-Hartenberg parameters are determined through the optimization of a penalty, which is positive when any design specification is violated and is zero when all the specifications are met. This is expressed mathematically as:

$$\min_{\text{DH}} \sum_{\text{task points}} \min_{\text{postures}} [\text{penalty\_function}(\text{DH}, (\theta_1, \ldots, \theta_n), \text{task}, \text{constraints})] \qquad (17)$$

where "penalty_function()" determines the penalty for the violation of the constraints, by a manipulator design, DH, in a posture, $(\theta_1, \ldots, \theta_n)$, reaching a specific task point.

By our definitions, a manipulator is first order fault tolerant with respect to a task if all possible first order ROD can fulfill the task. Equation (17) can then be adapted to include first order fault tolerance as follows:

$$\min_{\text{DH}} \sum_{\text{task points}} \left(\sum_{j=1}^{n} \max_{\theta_j} [\min_{\substack{\theta_i \\ (i \neq j)}} \text{penalty\_function}(\cdot)] \right) \qquad (18)$$

The maximization yields zero when all the ROD's, resulting from a failure of joint j, are able to reach the task point. As a result, the minimization over the D-H parameters equals zero if and only if all possible ROD's are able to reach every specified task point.

4.2 Implementation

Equation (18) contains three nested optimizations. For each ROD, the innermost minimization finds the posture, for which the penalty function is minimal. Since the task that we consider is to reach a certain point, perhaps the simplest penalty function one can use is the norm squared of the residue, $||r||^2$, namely, the distance between the actual position/orientation of the end effector and its desired position/orientation. The resulting minimization is thus a nonlinear least squares problem, and is commonly solved in robotics applications using a Newton-Raphson scheme [5]. This scheme converges very quickly when the residue at the optimum is zero, i.e., for a zero residual NLLS problem. However, it is possible that a task point is outside the reach of the manipulator, in which case the minimization becomes a large residual problem. For this class of problems, the Newton-Raphson scheme (or Gauss-Newton algorithm) converges slowly and is outperformed by the BFGS algorithm (Broyden-Fletcher-Goldfarb-Shanno [4]). As shown by Fletcher and Xu [3], the best overall performance is obtained by a hybrid algorithm, which combines the advantages of both the Gauss-Newton and the BFGS algorithm.

The second optimization in Equation (18) is the maximization over the joint angle, θ_j^f. A typical example of the objective function is shown in Figure 3. Notice that the function and its gradient are zero possibly over a large part of the domain, namely, where the task point is reachable. In order to solve the maximization, we need to incorporate additional information. The key observation is that when the desired point is outside the reach of the manipulator, the innermost minimization algorithm always gets stuck at the boundary of the workspace of the manipulator, where the Jacobian matrix of the manipulator loses rank [10]. The measure of isotropy, Δ, is defined as [6]

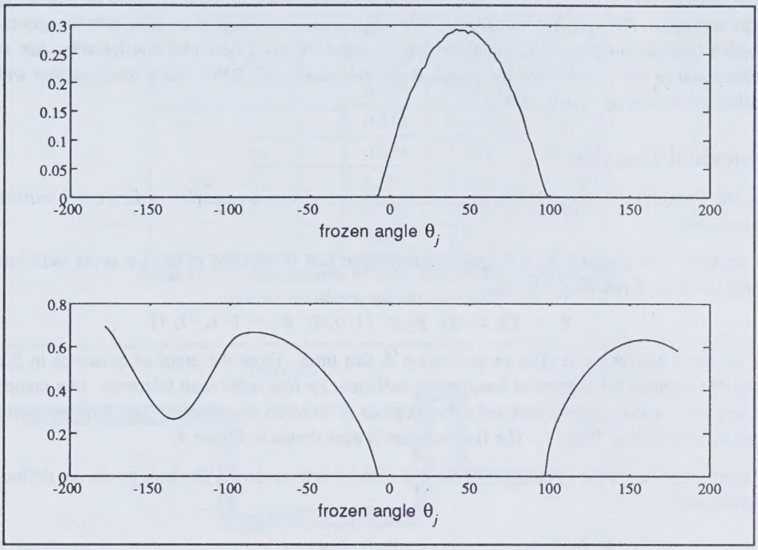


Figure 3: The objective function, $||r(\theta_j)||$, and the measure of isotropy.

$$\Delta = \frac{\sqrt[m]{\det(JJ^T)}}{\operatorname{trace}(JJ^T)/m} \tag{19}$$

with m the number of Cartesian coordinates. For a singular Jacobian, Δ is zero. Consequently, a minimization over Δ leads us to regions where the residue is possibly positive. Thus, the derivative of Δ with respect to θ_j^f, can be used to compensate for the lack of gradient information in the norm of the residue. However, as one can see in Figure 3, the function to optimize is multimodal. Since there is only one optimization variable, an efficient global optimization algorithm, such as the P^* algorithm developed by Zilinskas [15], can be used. P^* is a combination of a one-stage Bayesian algorithm and Brent's local minimization resulting in smart global coverage over the full domain and accurate local refinement.

The third and outermost optimization minimizes the total penalty of a kinematic configuration, defined by the $(3n-1)$ D-H parameters. Unfortunately, this function has multiple minima. Therefore, we need to employ a global optimization method, such as *simulated annealing*. Simulated annealing was first proposed as a combinatorial optimization algorithm [9], but can be easily adapted to continuously varying optimization variables. The method is basically a random iterative improvement algorithm with the modification that, under certain conditions, an *increase* in the objective function is accepted. A new trial configuration is generated randomly in the neighborhood of the current configuration. The condition for acceptance of this trial configuration is:

$$\begin{cases} \Delta F_{obj} \le 0 & \Rightarrow \text{accept} \\ \exp\left(-\Delta F_{obj}/T\right) > \text{random}\,[0,1\,) & \Rightarrow \text{accept} \end{cases} \tag{20}$$

which depends on a control variable, T, the temperature. The algorithm is started at a high temperature, for which all new configurations are accepted. After each iteration the temperature is decreased until no new acceptable configuration can be found. The search is then *frozen*. We adapted this basic algorithm to include the special properties of our objective function, in particular, the algorithm is stopped when a new trial configuration has an objective function value equal to zero, even when the search is not yet frozen. We know that a configuration with a penalty of zero satisfies all the design requirements.

5 Numerical Results

To illustrate the approach developed in the previous section, we give two examples of first order fault tolerant spatial manipulator designs.

In the first example, we design a 5-DOF spatial manipulator that is required to be first order fault tolerant while accomplishing the task of reaching 3 points:

$$\mathbf{p}_1 = (2, 2, -1) \quad \mathbf{p}_2 = (1, 0, 0) \quad \mathbf{p}_3 = (-1, -1, 1) \tag{21}$$

Notice that we don't consider end effector orientation at this time. From the proof of existence in Section 3, we know that for this example six degrees of freedom are sufficient for first order fault tolerance. Our numerical results show however, that for this specific task only five degrees of freedom are needed. The D-H parameters of three possible designs are listed in Table 1. The first solution is also shown in Figure 4.

A second example also takes the orientation of the end effector into account. The task points are defined as homogeneous transforms:

$$\mathsf{P}_1 = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & -1 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathsf{P}_2 = \begin{pmatrix} 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathsf{P}_3 = \begin{pmatrix} 0 & -1 & 0 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{22}$$

If the requirement of fault tolerance were dropped, a 6-DOF manipulator would be needed to fulfill this task. Duplication of each degree of freedom, as proposed in Section 3.1, would thus result in a 12-DOF fault tolerant manipulator which would be very much overdesigned. Indeed, the solutions found by our program show that nine degrees of freedom are sufficient for this task. Thus we have only 3 extra degrees of freedom instead of 6, and therefore the system is cheaper to build. We list the D-H parameters of three of our solutions in Table 2.

6 Conclusion

In this paper, we defined a property of a small class of redundant manipulators, called *fault tolerance*. Based on this definition, we were able to constructively prove the existence of fault tolerant manipulators by duplicating the joints. We also proved analytically that, in general, five degrees of freedom are needed for a planar manipulator to reach a set of position/orientations in a fault tolerant way. For spatial manipulators, we extended the design technique proposed in [13] to include fault tolerance as a design specification. This approach was illustrated with two examples of spatial fault tolerant manipulator designs.

solution	joint i	5 DOF solution (without orientation)		
		d_i	a_i	α_i
#1	1	2.242	0.787	83.5°
	2	1.011	0.427	-145.9°
	3	1.659	1.892	-110.1°
	4	0.580	0.438	-130.4°
	5	1.516	2.360	—
#2	1	0.406	0.000	-51.6°
	2	0.000	2.112	83.1°
	3	0.115	1.512	-54.7°
	4	0.000	0.599	-124.2°
	5	0.409	2.042	—
#3	1	0.314	0.614	44.7°
	2	0.804	1.558	-162.1°
	3	1.580	0.000	71.6°
	4	0.333	1.041	-129.7°
	5	0.920	1.594	—

Table 1: D-H parameters of three first order fault tolerant spatial solutions, without orientation.

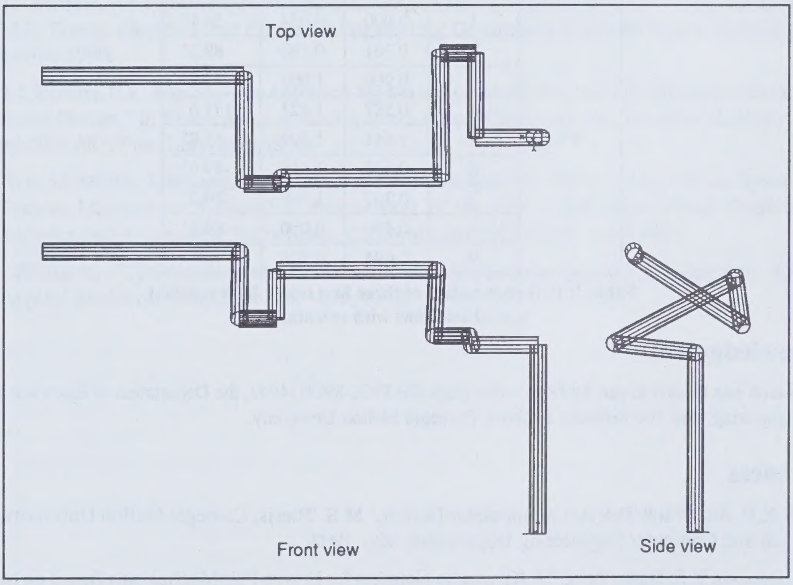


Figure 4: A 5 DOF spatial first order fault tolerant manipulator (without orientation).

solution	joint i	9 DOF solution (with orientation)		
		d_i	a_i	α_i
#1	1	0.566	0.917	-139.4°
	2	1.972	0.640	-132.4°
	3	1.085	1.622	-17.9°
	4	0.000	0.845	141.2°
	5	0.000	2.274	145.5°
	6	0.331	0.743	-160.3°
	7	1.523	0.252	65.8°
	8	0.242	0.747	-135.2°
	9	1.483	0.013	—
#2	1	0.000	0.922	-74.2°
	2	0.000	1.333	-137.1°
	3	0.448	0.481	17.7°
	4	0.168	1.121	158.4°
	5	0.902	1.323	-74.6°
	6	0.280	0.985	-137.8°
	7	0.000	0.088	-99.4°
	8	0.359	0.182	-127.7°
	9	0.610	0.423	—
#3	1	0.000	0.054	56.1°
	2	0.701	0.160	-89.2°
	3	0.906	1.003	-77.8°
	4	0.587	1.421	171.6°
	5	1.641	1.399	55.8°
	6	0.662	1.418	-49.9°
	7	0.267	1.969	-49.2°
	8	0.496	0.000	-89.8°
	9	0.694	0.526	—

Table 2: D-H parameters of three first order fault tolerant spatial solutions with orientation

Acknowledgment

This research was funded in part by DOE under grant DE-F902-89ER14042, the Department of Electrical and Computer Engineering, and The Robotics Institute, Carnegie Mellon University.

References

[1] W.K.F. Au, “Fault Tolerant Manipulator Design,” M.S. Thesis, Carnegie Mellon University, Electrical and Computer Engineering Department, May 1992.

[2] J. Denavit, R.S. Hartenberg, “A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,” *Journal of Applied Mechanics*, Vol. 22, No. 2, pp. 215–221, June 1955.

[3] R. Fletcher, C. Xu, "Hybrid Methods for Nonlinear Least Squares," *IMA Journal of Numerical Analysis*, Vol. 7, No. 3, pp. 371–389, July 1987.

[4] R. Fletcher, *Practical Methods of Optimization, Second Edition*, John Wiley & Sons, New York, 1987.

[5] P.K. Khosla, C.P. Neuman, F.B. Prinz, "An Algorithm for Seam Tracking Applications," *The International Journal of Robotics Research*, Vol. 4, No. 1, pp. 27–41, Spring 1985.

[6] J.-O. Kim, P.K. Khosla, "Dexterity Measures for Design and Control of Manipulators," in *Proceedings of IROS'91: International Workshop on Intelligent Robots and Systems*, pp. 758–763, Osaka, Japan, Nov. 3–5, 1991.

[7] J.-O. Kim, P.K. Khosla, "A Multi-Population Genetic Algorithm and Its Application to Design of Manipulators," in *Proceedings of IROS'92: International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 7–10, 1992.

[8] J.-O. Kim, *Task Based Kinematic Design of Robot Manipulators*, Ph.D. thesis, Robotics Ph.D. Program, Carnegie Mellon University, Pittsburgh, PA, August 1992.

[9] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671–680, May 1983.

[10] D. Kohli, M.-S. Hsu, "The Jacobian Analysis of Workspaces of Mechanical Manipulators," *Mechanism and Machine Theory*, Vol. 22, No. 3, pp. 265–275, 1987.

[11] A.A. Maciejewski, "'Fault Tolerant Properties of Kinematically Redundant Manipulators," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, Ohio, pp. 638–642, May 1990.

[12] C.J.J. Paredis, *An Approach for Mapping Kinematic Task Specifications into a Manipulator Design*, M.S. Thesis, Electrical and Computer Engineering Department, CarnegieMellon University, September 1990.

[13] C.J.J. Paredis, P.K. Khosla, "An Approach for Mapping Kinematic Task Specifications into a Manipulator Design," in *Proceedings of the 5th International Conference on Advanced Robotics*, Vol. 1, pp. 556–561, Pisa, Italy, June 1991.

[14] E. Wu, M. Diftler, J. Hwang, J. Chladek, "A Fault Tolerant Joint Drive System for the Space Shuttle Remote Manipulator System," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, pp. 2504–2509, April 1991.

[15] A. Zilinskas, "Optimization of One-Dimensional Multimodal Functions, Algorithm AS 133," Applied Statistics, Vol. 23, pp. 367–375, 1978.

Computing a Statistical Distribution of Stable Poses for a Polyhedron[*]

Jeff Wiegley[†] Anil Rao[‡] and Ken Goldberg[§]
Institute of Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273.

September 1992

Abstract

In high-volume manufacturing, it is important to predict how parts will behave as they proceed through the assembly line. In this paper we describe a method for estimating the distribution of stable poses in a stream of identical polyhedral parts. We assume that each part's initial orientation with the worksurface is a uniform random variable and that the part falls under quasi-static conditions (*i.e.* without dynamics). For an n-sided polyhedron, our algorithm runs in time $O(n^2)$.

1 Introduction

As it is extremely costly to maintain part order throughout the manufacture cycle, for example by keeping parts in pallets, parts are often delivered in bags or boxes where they must be fed to an assembly station.

> *In [robot] assembly, two-thirds of the system costs are typically for peripherals such as supporting fixtures, special grippers, and part presentation devices. These peripherals are generally neither versatile nor programmable.* [3]

Whitney [10] noted that although parts handling accounts for a large part of total assembly time, there is currently little scientific basis for the process of recognizing, positioning, and orienting parts. The manufacturing industry currently uses a variety of ad-hoc methods. In discussions with six professional manufacturing engineers, the term "black magic" came up repeatedly when they described the process of customizing a parts feeder for a new part.

---

[*] This research was supported by a Cooperative Agreement between the National Institute for Standards and Technology (NIST) and the Institute for Manufacturing and Automation Research (IMAR), and by equipment grants from Adept Technology, Inc., PhD, Inc., and the Robotics and Automation Corporation.

[†] Department of Computer Science, `wiegley@usc.edu`.

[‡] Department of Electrical Engineering–Systems, `rao@halcyon.usc.edu`, 213-740-2350.

[§] Departments of Computer Science and Electrical Engineering–Systems, `goldberg@usc.edu`, 213-740-9080.

> *The smartest assembly robot and the best assembly machine in the world are useless without the mechanism that delivers the part.* [5].

Our group at USC is studying methods for handling industrial parts [8, 6, 15, 2, 14, 7]. In our previous work we assumed that part motion is planar (ie, that parts are already in contact with a smooth flat worksurface). In this paper we consider how parts will arrive on the worksurface. In particular, we consider the statistical distribution of poses for a stream of identical polyhedral parts that are randomly dropped under quasi-static conditions [12]. We acknowledge that under true dynamic conditions, the statistical distribution or poses can be quite different. However, it is notoriously difficult to model the dynamics of collisions with friction [9].

1.1 Related Work

To compute probabilities we use a method similar to that presented in [1] to compute the initial contact distribution. We show that the final resting distribution will be different if some faces of the convex hull are unstable, and give one method for resoving these differences. Also, we report on the computational complexity of our algorithm.

Erdmann, Mason and Vanecek [4] reported a computational approach to orienting three-dimensional parts. Given an n-sided polyhedral part resting on a planar table, the objective is to find a sequence of tilting angles for the table that will bring a particular part face into contact with the table (thereby eliminating all but one degree of rotational freedom). Assuming that friction is infinite and that the part never rolls off the edge of the table, the authors showed how to construct such a plan, if one exists, in time $O(n^4)$. Their method does not depend on the initial distribution of part poses.

Palmer [11] considered the complexity of modelling mobility and stability for an arrangement of (2D) polygons. He found the following two (seemingly straightforward) problems to be NP-Hard: (1) determining whether one or more polygons in the set can be moved without causing intersection, (2) determining if the arrangement is at a static equilibrium point. Although in our case we deal with 3D polyhedra, we only consider a single part rather than an arrangement.

2 The Algorithm

Assumptions

- The part is a convex polyhedron with a known center of mass
- The worksurface is smooth, flat, and much larger than the part.
- The orientation of the part upon contact is a uniform random variable on $SO(3)$.
- The part's motion is quasi-static.

Given

P_0 : a wireframe model of an n-sided polyhedral part.

M : P's center of mass.

The algorithm

1. Compute the convex hull of the polyhedral part, call it P.

   Note that it is sufficient to analyze the convex hull of P_0 since any stable pose will correspond to a face of P resting against the worksurface. [13] give a variety of methods for computing convex hulls; a divide-and-conquer strategy can be used to compute the convex hull in time $O(n \log n)$. Here we use the gift-wrapping method to compute it in time $O(n^2)$.

2. For each face of the convex hull, compute the initial probability. Let p_i be the probability of landing on face i

   For this step we essentially follow Boothroyd's method. Since all faces of P are convex, it is sufficient to project each face out from the center of mass onto a unit sphere centered at M.. The relative areas of projection give the relative probabilities. Let $\beta_1, \ldots, \beta_k$ be the radian measures of the interior angles of face i, the relative area for the projected face is:

   $$A_i = \frac{\sum_{j=1}^{k} \beta_j - (n-2)\pi}{4\pi} \tag{1}$$

   To compute the internal angles, let A', B', C' be three vertices of the part and A, B, C be the projected points on the unit sphere. To compute $\beta = \angle \mathrm{ABC}$, let l_1 be the line passing through B perpendicular to AM and l_2 be the line passing through B perpendicular to CM. It is easy to verify that the angle between these two lines at B is β.

3. To handle the unstable faces of P, we build a directed acyclic graph G.

   Let G have n vertices corresponding to the faces of P. Attach a directed arc $(\overrightarrow{i,j})$ if the polygon sitting on face i will topple (quasi-statically) onto face j. To decide if a face will topple, we project M onto the plane of face i. Let the projected point be M'. We have identified 5 possible types of stability for a face:

   (a) Stable: M' lies inside the face. P will not topple further.

   (b) Non-stable, one edge: M' lies outside the face and there is one unique maximum moment. In this case P will topple over that edge.

   (c) Non-stable, two edges: M' lies outside the face and is equidistant from two edges that share a common vertex. In this case we assume P is equally likely to topple over either edge.

   (d) Semi-stable, one edge: M' lies above an edge of the face but not at a vertex. We assume that the part is equally likely to remain stable or topple over that edge.

   (e) Semi-stable, two edges: M' lies above an edge of the face at a vertex. We assume that the part is equally likely to remain stable or topple over either edge.

4. To propagate probabilities to stable final faces, we traverse the graph breath first, pushing the resting probabilities along the edges of G.

2.1 Computational Complexity

The number of vertices and edges of P are both $O(n)$. as stated above, we compute the convex hull in time $O(n^2)$. We compute the projected areas in $O(n^2)$ time, since for each face we need to compute $O(n)$ interior angles. The DAG can be built in (n^2) since for each face the center of mass must be checked against $O(n)$ edges. Finally to traverse the DAG we are required to visit each node (face) only once so this requires $O(n)$ time. The total time then is thus $O(n^2)$.

3 Implementation

The algorithm has been implemented in C using Motif to build a graphical interface so that we can display all of the stable poses.

3.1 Examples

The figures 1, 2, 3 illustrate the algorithm using an auto part.

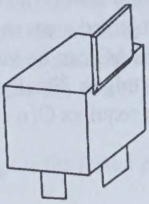


Figure 1: The automobile relay.

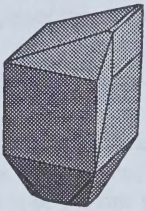


Figure 2: The convex hull of the automobile relay.

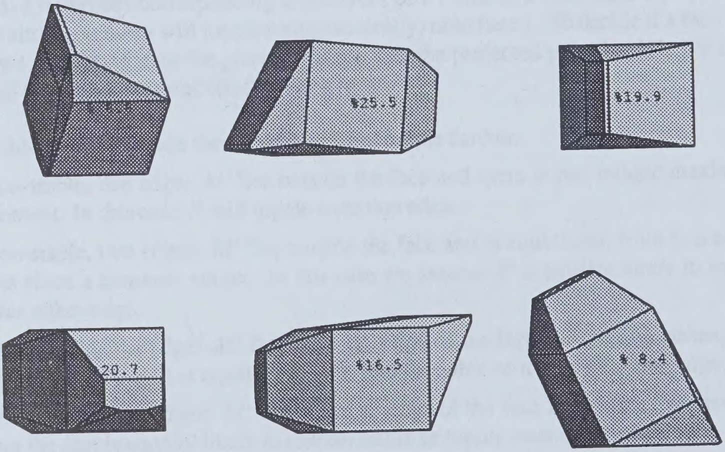


Figure 3: The stable poses of the relay with associated probabilities.

Another another example is a typical chemistry lab clamp of the type to hold two rods a specified angle to each other. Figures 4, 5 and 6 illustrate.

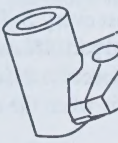


Figure 4: Chemistry clamp used for study.

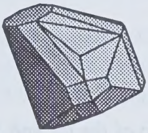


Figure 5: Chemistry clamp's convex hull.

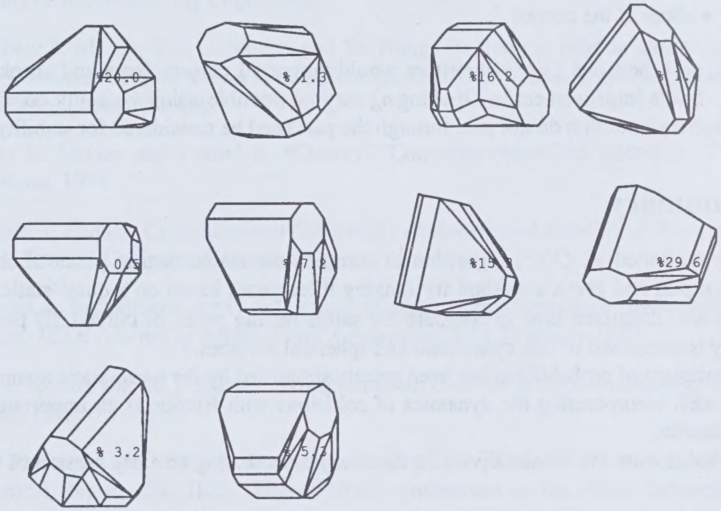


Figure 6: Resulting stable poses and probabilities for the chemistry clamp

4 Computing Stable Poses for a Class of Curved Parts

Let us say that a part is a "generalized polyhedron" if its boundary consists of flat, spherical, and cylindrical surfaces. Given a convex generalized polyhedron, we can determine its stable poses as follows. Let the part's surfaces be $S_1, S_2, \ldots, S_n$, where each S_i is the portion of a plane, sphere, or cylinder. By "a portion of," we understand that the characteristic function for a surface S_i is computable: given a point p, $f_i(p) = 1$ if p on S_i (*i.e.* p lies on a plane/sphere/cylinder of the part) and 0 otherwise.

Given the part's center of mass, we test every S_i for stability as follows. If S_i is a (portion of) plane, treat it as we did for a polyhedral part. If it is part of a sphere, the only possible stable position is when the point on the sphere closest to the c.o.m. is in contact with the surface. If it is part of a cylinder, the only stable pose is when the closest edge of the cylinder to the c.o.m. is in contact with the surface.[1]

If we are given a non-convex generalized polyhedron, we can identify all stable poses where the part is in contact with 3 or more points by noting that the contact must be one of the following:

1. a face of the original part
2. a tangent planes:
    - between one vertex and two of the curved S_i
    - two vertices and one of the S_i
    - an edge and one of the S_i
    - three of the curved S_i

Thus, an exhaustive $O(n^3)$ algorithm would output all tangent faces and check each for stability. Some improvement to $O(n^2 \log n)$ may be possible using visibility considerations (only tangent planes that do not pass through the part need be considered for stability).

5 Summary

We have presented an $O(n^2)$ algorithm to compute the stable resting poses of an n-sided polyhedral part and given a method for ranking these poses based on a quasi-static analysis. We have also discussed how to compute the stable resting poses of curved 3D parts whose boundary is composed of flat, cylindrical and spherical surfaces.

Our estimate of probabilities has been greatly simplified by the quasi-static assumption. A better model, incorporating the dynamics of collisions with friction is an important area for future research.

Acknowledgments We thank Olivier Tardieu for implementing an early version of this algorithm.

References

[1] G. Boothroyd, A. H. Redford, C. Poli, and L. E. Murch. Statistical distributions of natural resting aspects of parts for automatic handling. *SME Manufacturing Engineering Transactions*, 1:72, 1972.

---

[1] In case of spheres and cylinders, there could be infinite stable poses. This happens when the c.o.m. lies at the center (axis) of the sphere (cylinder). This condition can be easily checked.

[2] John Canny and Ken Goldberg. A risc paradigm for robotics. August 1992. (Submitted to IEEE International Conference on Robotics and Automation).

[3] Peter Dewhurst. Design for robotic assembly. *Manufacturing Engineering*, June 1991.

[4] Michael Erdmann, Matthew T. Mason, and George Vanecek Jr. Mechanical parts orienting: The case of a polyhedron on a table. In *International Conference on Robotics and Automation*. IEEE, April 1991.

[5] Gregory T. Farnum and Bill Davis. Delivering the part. *Manufacturing Engineering*, March 1986.

[6] Ken Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 1992 (to appear).

[7] Ken Goldberg, Matt Mason, and Ari Requicha. Nsf workshop on geometric uncertainty in robot motion planning: Summary report and bibliography. Technical Report 297, IRIS, August 1992. (The NSF-sponsored Workshop was held on Catalina Island, June 14-17, 1992).

[8] Ken Y. Goldberg. Handling industrial parts with the parallel-jaw gripper. In *Design and Manufacturing Grantees Conference*, January 1993. (Proceedings published by the Society of Manufacturing Engineers).

[9] Matthew T. Mason, Ken Goldberg, and Yu Wang. Progress in robotic manipulation. In *15th Grantees Conference on Production Research and Technology*. National Science Foundation, January 1989.

[10] James L. Nevins and Daniel E. Whitney. Computer-controlled assembly. *Scientific American*, 1978.

[11] Richard S. Palmer. *Computational Complexity of Motion and Stability of Polygons*. PhD thesis, Cornell Computer Science, 1989.

[12] Michael A. Peshkin and Art C. Sanderson. Minimization of energy in quasi-static manipulation. *IEEE Journal of Robotics and Automation*, 5(1), February 1989.

[13] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.

[14] Anil Rao and Ken Goldberg. Grasping curved planar parts with a parallel jaw gripper. Technical Report 299, IRIS, August 1992. (submitted to the IEEE Transactions on Robotics and Automation).

[15] Anil Rao and Ken Goldberg. On the relation between friction and part curvature in parallel-jaw grasping. Technical Report 298, IRIS, August 1992. (submitted to the 1993 IEEE International Conference on Robotics and Automation).

Machine Vision Systems for Shape Recovery

Shree K. Nayar

Department of Computer Science
Columbia University
New York, N.Y. 10027

Abstract

The advancement of three-dimensional machine vision is closely related to the development of robust and efficient shape recovery methods. We address the recovery problem associated with three different classes of surfaces: (a) specular surfaces; (b) surfaces with varying reflectance; and (c) rough and textured surfaces. For specular surfaces, we have developed the structured highlight technique that uses a large array of point sources to recover surface orientation from image highlights. The sources are efficiently scanned using a binary coding scheme. For surfaces with spatially varying reflectance properties, the photometric sampling method is presented. This method uses a small array of extended sources to compute local estimates of surface orientation and reflectance, simultaneously. Photometric sampling can adapt to reflectance variation from specular to diffuse over the object surface. Finally, we address the recovery problem associated with rough and textured surface. We show that focus analysis provides an effective means of extracting the shapes of surfaces with complex roughness and reflectance properties. Three real-time machine vision systems have been developed based on these results. Experimental results demonstrate that the proposed methods and systems are applicable to a variety of visual inspection problems.

1 Introduction

Automatic inspection systems are vital to the advancement of manufacturing technology and quality control. Today, most real-time inspection systems use binary image processing techniques and pattern matching methods to compare inspected objects with their models. There are however several industrial problems that require the inspection of the three-dimensional shapes of manufactured parts. The advancement of three-dimensional machine vision is largely dependent on the development of robust and efficient shape recovery methods.

Machine vision research has produced several shape recovery techniques such as binocular stereo, structured light, photometric stereo, and shape from shading. Though these techniques are novel in their approaches, they each have limitations that often preclude them from being viable solutions to real-world problems. The current status of vision research suggests that there is no single method that can be applied to all types of surfaces. Surfaces are characterized by not only their shapes but also their reflectance and roughness properties. These properties may vary from one inspection task to another. Existing recovery methods are often incapable of adapting to substantial variations in surface reflectance. Further, there are types of surfaces that cannot be handled by any of these vision techniques.

On the assembly line, several difficult inspection tasks are still performed by human inspectors. We know that the human vision system is powerful and versatile; it can perceive the shapes and reflectance properties of a wide range of surfaces. However, there are surfaces, such as microscopic surfaces, that are difficult to perceive even for the human eye. For some of these problems, machine vision can provide effective solutions. By focusing on a particular problem, it is possible to develop machine vision systems that out-perform humans in both accuracy and speed.

In this paper, we present three shape recovery methods. These recovery methods are based on physical models of surface reflection and image formation. Each method has been developed to recover the shapes of a particular class of surfaces. The surface classes examined here are (a) specular surfaces; (b) surfaces with varying reflectance, and (c) rough and textured surfaces. In the following sections, we describe the problems posed by these surfaces and the vision techniques we have developed to solve them. These results indicate that the proposed shape recovery methods can be used to solve several challenging visual inspection tasks.

2 Specular Surfaces

Surfaces of machined or polished metal parts and solder joints are examples of specular surfaces. For a purely specular surface, light is reflected such that the angle of incidence equals the angle of reflection. Illumination of a specular surface using a point source of light does not produce smooth shading on the surface. Fig. 1 shows a camera image of several specular objects. Images of such objects are difficult to interpret as they are characterized by bright points or ***highlights***, and inspection of surface shape is a challenging task.

The structured highlight technique uses a large array of point sources to illuminate the inspected object [Sanderson, Weiss, & Nayar 88] [Nayar et al. 90]. The point sources are uniformly distributed around the object, and images of the object are obtained using a cam-

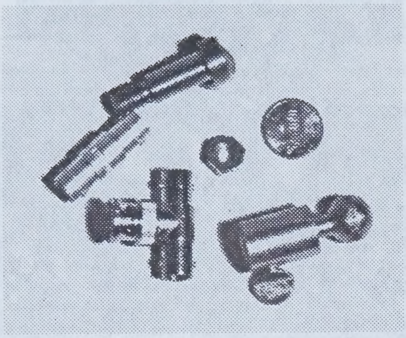


Figure 1: Image of several specular objects.

era. The point sources are activated and highlights on the object surface are used to compute local surface orientations. The sequential illumination of the sources is time consuming and would make it impossible to use the method for real-time inspection tasks. To overcome this problem, we have developed a binary coding scheme that activates selected sets of sources to obtain a small set of images. A prototype structured highlight inspection system, called SHINY, has been implemented and used to extract the shapes of specular objects including solder joints.

2.1 Structured Highlight

Consider the illumination and imaging geometry shown in Fig. 2. A surface element is located at the origin of a viewer-oriented coordinate frame and is illuminated by a point source of light. The source vector **s** is determined by the position of the source and the sensor vector **v** is determined by the location and image projection model of the camera. In the case of specular reflectance, the surface element has to be properly oriented to reflect light from the point source into the camera. If the point source is moved around the surface element, a highlight will be detected in the camera image at some source location. If the source vector **s** is known at the point of highlight detection, the normal (orientation) vector **n** of the surface element can be determined using the specular constraint that requires the source, sensor, and normal vectors be coplanar and the angle of reflection be equal to the angle of incidence:

$$\mathbf{n} = \frac{\mathbf{v} + \mathbf{s}}{\| \mathbf{v} + \mathbf{s} \|} \qquad (1)$$

Structured highlight uses a large array of point sources to determine local surface orientations of specular surfaces. The estimation of orientations is based on the *distant source assumption*; the distance of each source from the origin of the viewer-oriented frame is assumed to be large compared to the distance of surface points from the origin. Under this assumption, the direction of any particular source is the same for all points on the object surface, and is determined only by the position of the source in the viewer-oriented frame.

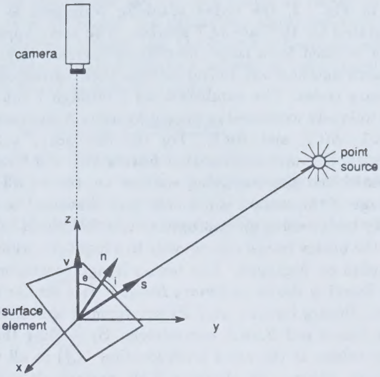
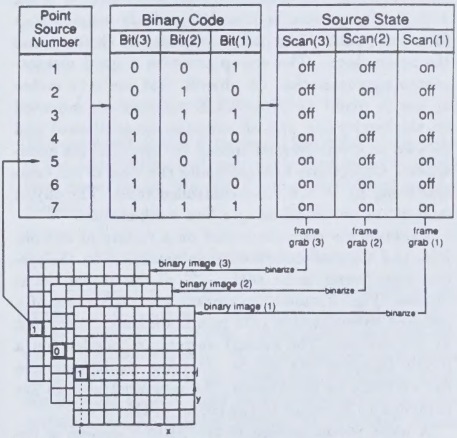


Figure 2: Surface element at the origin of a viewer-oriented frame.

Consider a specular surface illuminated by two point sources from two different directions. The resulting image of the object may contain several highlights. Since the shape of the surface is not known, it is impossible to determine which of the two sources produced any particular highlight. This information, however, is necessary for the estimation of surface orientation. The simplest way to overcome this problem is by activating the point sources one at a time, obtaining an image of the object for each source, and computing orientations from the highlights. This approach is referred to as *sequential scanning*. In our implementation, we have used an alternative approach where the point sources are scanned using coded patterns to improve the efficiency of the scanning process. For a large array of sources, *coded scanning* is far more efficient than sequential scanning.



Point Source Number	Binary Code			Source State		
	Bit(3)	Bit(2)	Bit(1)	Scan(3)	Scan(2)	Scan(1)
1	0	0	1	off	off	on
2	0	1	0	off	on	off
3	0	1	1	off	on	on
4	1	0	0	on	off	off
5	1	0	1	on	off	on
6	1	1	0	on	on	off
7	1	1	1	on	on	on

Figure 3: An example of point source scanning using the binary coding scheme. Seven point sources are scanned using three illumination patterns.

In Fig. 3, the coded scanning technique is illustratated for the case of 7 sources. The same approach can be used for a larger number of sources. First, the source numbers are converted into their corresponding binary codes. The numbers from 1 through 7 can each be uniquely expressed in binary by using 3 bits, namely, *Bit* 1, *Bit* 2, and *Bit* 3. For the first scan, namely, *Scan* 1, all point sources that have a high *Bit* 1 are activated and the remaining sources are turned off. An image of the surface is obtained and converted to a binary image using an appropriate threshold level. A "1" in the binary image corresponds to a highlight while "0" implies no highlight. The binary image corresponding to *Scan* 1 is shown as *Binary Image* 1. In a similar manner, *Binary Image* 2 and *Binary Image* 3 are obtained for *Scan* 2 and *Scan* 3, respectively. By reading the image values at the same pixel location (i,j) in all three binary images, we obtain a 3-bit pattern; (1,0,1) for the example considered. We assume that, due to high surface specularity, only a single source can generate a highlight at any particular point on the surface. Therefore, the bit pattern (1,0,1) could result only if the surface point corresponding to image point (i,j) reflected light from source 5 into the camera. The surface orientation at point (i,j) is computed by using the known direction of source 5, the viewing direction of the camera, and the specular reflectance model. This coding scheme can be generalized such that $2^N - 1$ point sources can be scanned using N illumination patterns. In our implementation of the structured highlight method, we have used a total of 127 sources and therefore 7 images of the object are needed to compute a complete orientation map.

2.2 SHINY System

The Structured Highlight Inspection System (SHINY) was implemented [Nayar et al. 90] to demonstrate the practical feasibility of the structured highlight method. The system consists of a hemispherical array of 127 point sources. Each point source is generated using a 1000 milli-candela light emitting diode coupled to a fiber-optic cable. The object is placed at the center of the hemisphere. The set-up provides a good distant-source approximation for objects that are 0.12 inches or less in diameter. Four CCD cameras are mounted on the hemisphere and all cameras are positioned and focused on a small region around the center of the hemisphere. Objects are brought under the view of the cameras using an X-Y-θ servo-controlled table. The object images are processed using a Sun work-station.

Experiments were conducted on a variety of test objects and the surface orientations measured by the system were found to be within 3% of the actual orientations. Fig. 5 shows the surface normal vectors of a polished sphere that is 1/16 inch in diameter, measured by the system. The normal vectors are overlaid on a profile image of the sphere. In order to demonstrate the accuracy of the system, the measured vectors are extended to intersect at the center of the sphere.

A more recent version of the SHINY system is being used by Westinghouse Electic Corporation to inspect solder joints on surface-mount circuit boards. A photograph of the orientation measuring device used by this SHINY system is shown in Fig. 4. The reflectance model of a typical solder surface has a dominant specular component. This renders solder inspection a suitable candidate for structured highlight inspection. The measured orientations of each solder joint are used to generate an Extended Gaussian Image (EGI). The EGI is a compact and powerful representation for object shape based on surface orientation information [Horn 84]. Several features are extracted from the EGI, and solder joints are classified based on the feature values. Using customized hardware, the Westinghouse system is able to inspect solder joints at a rate of 2 joints per seconds. A typical surface-mount board (with about 2000 joints) can be inspected by the SHINY system in approximately 18 minutes. A human inspector could take anywhere from 60 to 90 minutes to inspect the same board.

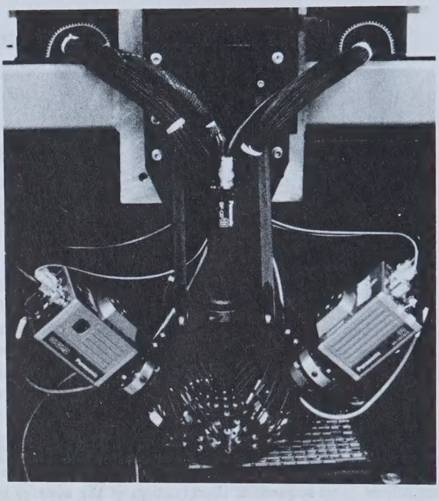


Figure 4: The structured highlight inspection system (SHINY) developed to recover the shapes of specular surfaces.

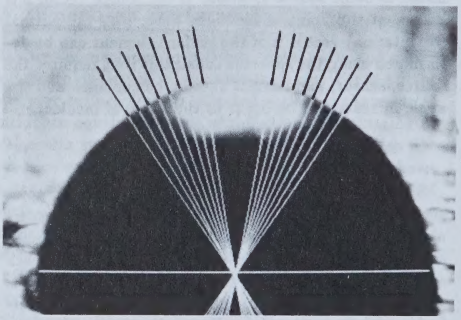


Figure 5: Surface normal vectors of a polished sphere computed using the SHINY system.

3 Surfaces with Varying Reflectance

The structured highlight method is effective and efficient only for the class of specular surfaces. Surfaces that have a significant diffuse component of reflection produce shading or brightness variation, and image intensity values must be analyzed to recover their shapes. Shape from shading [Horn 70][Pentland 84] and photometric stereo [Woodham 78][Coleman & Jain 82] are examples of techniques that extract three-dimensional shape information from image intensities. All of these techniques are applicable to diffuse surfaces but they rely on prior knowledge of the reflectance model of the surface. Further, these methods assume that all points on the object have the same reflectance properties. The method by Coleman and Jain [Coleman & Jain 82] can handle surfaces with diffuse and specular components but it relies on the existence of a significant diffuse component; it cannot be used to recover surface regions that are primarily specular. Many objects encountered in practice consist of surface regions that have very different reflectance properties. Fig. 6 shows several objects with regions of different reflectance characteristics; diffuse and specular. The above shape recovery methods are not applicable to objects such as these.

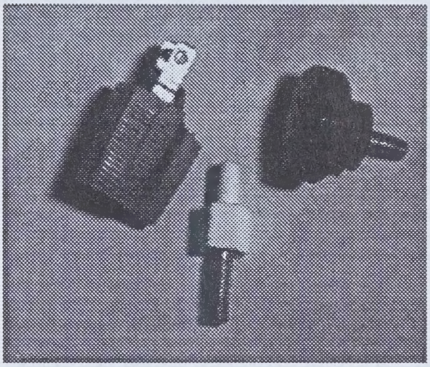


Figure 6: Objects with surface regions of different reflectance properties.

In this section, we present *photometric sampling* [Nayar, Ikeuchi, & Kanade 90], a shape-from-intensity method that can adapt to variations in surface reflectance. Reflectance invariance is achieved by computing both shape and reflectance, simultaneously. We use a hybrid reflectance model that is a linear combination of diffuse and specular components. The relative strengths of the diffuse and specular components are unknown and can vary from one surface point to the next. Ikeuchi [Ikeuchi 81] demonstrated that extended light sources are needed to obtain measurements from a continuous range of orientations on a specular surface. Based on this observation, an illumination technique is proposed that uses an array of extended sources. The source directions are selected by sampling the illumination space in a systematic manner. The sources are activated one by one and for each source an image of the object is obtained. An extraction algorithm uses the set of object images and the hybrid reflectance model to estimate both orientation as well as reflectance parameters at each surface point.

3.1 Photometric Sampling

Consider the illumination of an object by a point source of light, as shown in Fig. 7. Light energy reflected by the surface in the direction of the camera causes an image of the surface to be formed. We assume the surface to be hybrid in reflectance; it has a diffuse and a specular component. The diffuse component is represented using the Lambertian model. The surfaces are assumed to be smooth. Hence, the specular intensity component is very sharp function of the source direction and may be approximated by the delta function. The *basic photometric function* relates image intensity to surface orientation, surface reflectance, and point source direction:

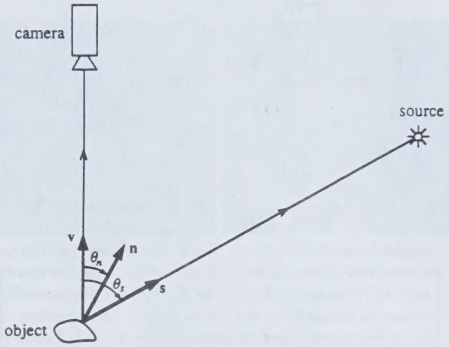


Figure 7: Two-dimensional illumination and imaging geometry. A surface element with orientation θ_n reflects light from the point source direction θ_s into the camera.

$$I = A\,cos(\theta_s - \theta_n) + B\,\delta(\theta_s - 2\theta_n) \qquad (2)$$

The constants A and B represent the relative strengths of the Lambertian and specular components of reflection, respectively. We call A and B the *reflectance parameters.*

3.1.1 Extended Light Sources

The photometric sampling method uses a set of extended light sources to illuminate the surface. Extended sources are chosen, rather than point sources, for the following reason. To detect specular reflections from surface points of all orientations, an infinite number of point sources need to be positioned around the surface. Unlike a point source, an extended source emits light from an area rather than a single point. Therefore, a small number of extended sources may be used to ensure the detection of specular reflections.

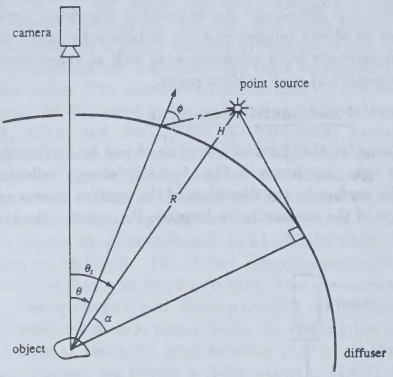


Figure 8: An extended light source.

Fig. 8 shows the technique we have used to generate extended sources. We assume that the diffuser is "ideal", i.e. any point on its surface appears equally bright from all directions. Then, the radiance of the inner surface of the diffuser is proportional to the irradiance of the outer surface. The direction of the extended source is denoted by the angle θ_s, i.e. the direction of the point source used to generate the extended source. The radiance function $L(\theta, \theta_s)$ is symmetric, or even, with respect to the source direction ($\theta = \theta_s$), and its magnitude decreases as θ deviates from θ_s. The expression for the extended source radiance is given in [Nayar, Ikeuchi, & Kanade 90] and is derived using the geometry shown in Fig. 8.

The photometric function for point source illumination (equation 15) needs to be modified for extended source illumination. An extended source may be thought of as a collection of point sources in which each point source has a radiant intensity that is dependent on its position on the extended source. Hence, the modified photometric function is determined by convolving the basic photometric function with the extended source radiance function $L(\theta, \theta_s)$. The result is [Nayar 90]:

$$I' = A' \cos(\theta_s - \theta_n) + B' L(2\theta_n, \theta_s) \tag{3}$$

Note that the Lambertian component remains a cosine function of the angle of incidence while the specular component is now represented by the extended source radiance function. The parameters A' and B' are proportional to the reflectance parameters A and B, respectively. In Fig. 9 image intensity I' is plotted as a function of extended source direction θ_s.

3.1.2 Sampling the Photometric Function

The process of measuring image intensities for different source directions is equivalent to sampling the modified photometric function $I'(\theta_s)$ (Fig. 9). Samples of the photometric function may be obtained by distributing an array of extended sources around the object. The entire array is sequentially scanned such that, for each scan, a single source is active and an image of the object surface is obtained. The sequential scanning of sources in the directions $\{\theta_i$: i=1,2,......M$\}$ results in a set of image intensities $\{I'_i$: i=1,2,......M$\}$ measured at each point on the object surface (Fig. 9). The number of intensities measured is determined by the frequency at which $I'(\theta_s)$ is sampled. In our implementation of the photometric sampling method we have chosen a total of 8 light sources. The reasons for this choice are given in [Nayar 90].

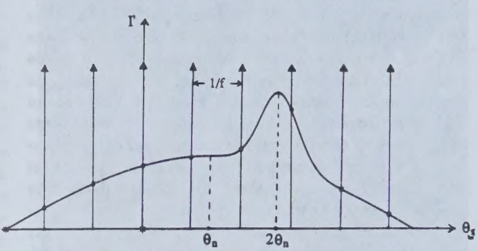


Figure 9: Sampling the modified photometric function.

3.1.3 Extracting Shape and Reflectance

The extraction problem is stated at follows: at each surface point, compute the orientation θ_n and the reflectance parameters A' and B' from the set of image intensities $\{I'_i$: i=1,2,......M$\}$ and the source directions $\{\theta_i$: i=1,2,......M$\}$. Equation 3 is non-linear in the unknown parameters A', B', and θ_n. Hence a closed-form solution for the parameters is difficult to obtain. We have developed an algorithm that first separates the Lambertian and specular components of each measured intensity value. It then computes orientation and reflectance parameters using the Lambertian and specular models.

A detailed description of the extraction algorithm is given in [Nayar, Ikeuchi, & Kanade 90]. It is not only applicable to hybrid surfaces that have both Lambertian and specular components, but also ideal Lambertian and ideal specular surfaces. Also noteworthy is the fact that the algorithm is local; orientation and reflectance parameters of a surface point are computed only from image intensities measured at that point. As a result, no assumptions need be made regarding the smoothness or continuity of the surface. Further, the reflectance parameters A' and B' can vary from one point on the object surface to the next.

3.2 Photo-Sampler System

A photometric sampling system was developed to recover the shape and reflectance of surfaces (Fig. 10). A 14-inch diameter lamp shade is used as a spherical diffuser and extended light sources are generated on the diffuser's surface by illuminating it using incandescent light bulbs. A total of 8 extended sources are generated in this manner. Each extended source has an angular width of 64 degrees, and adjacent sources are separated by an angle of 32 degrees. The object is placed at the center of the diffuser and is viewed by a CCD camera through a 1-inch diameter hole at the top of the diffuser. In this implementation, the light bulbs, camera, and object are all placed in the same plane. This two-dimensional set-up is capable of measuring only surface normal vectors that lie on a single plane in orientation space. For each extended source, an image of the object is digitized and stored in memory. The sequence of object images generated by scanning the array of extended sources is processed on a Sun work-station.

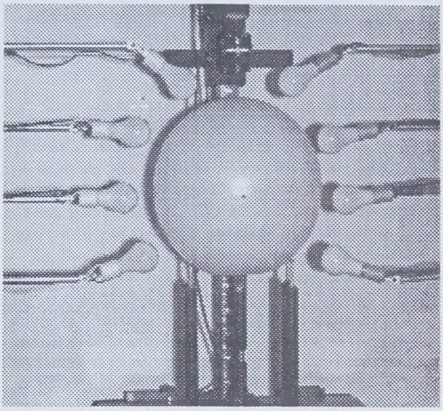
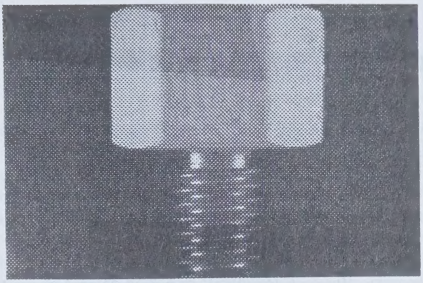


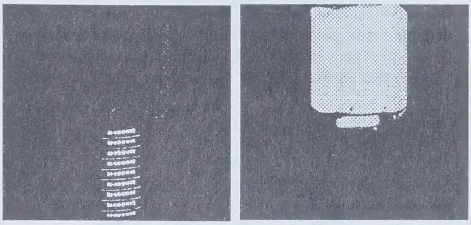
Figure 10: The photo-sampler system.

The photo-sampler and the extraction algorithm were used to compute the shape and reflectance of a variety of objects, including, matte painted objects, smooth metal objects, and plastic objects [Nayar, Ikeuchi, & Kanade 90]. Fig. 11 shows a metal bolt with two regions of different reflectance properties. The painted surface of the head is diffuse in reflection, while the threaded section of the bolt is specular. Figs. 11b and 11c show the reflectance images (Lambertian and specular) computed by the extraction algorithm. The intensities in the Lambertian and specular reflectance images, are proportional to the computed Lambertian albedo (A') and specular albedo (B'), respectively. Fig. 11d shows the needle map computed by the extraction algorithm. The needle map is a representation of surface orientations. At each point on a needle map, the length of the needle is proportional to the slant of the surface away from the viewing direction of the camera. An error analysis was conducted to estimate the accuracy of the computed normals. The mean of the absolute error in computed surface orientations was found to be less than 2 degrees.

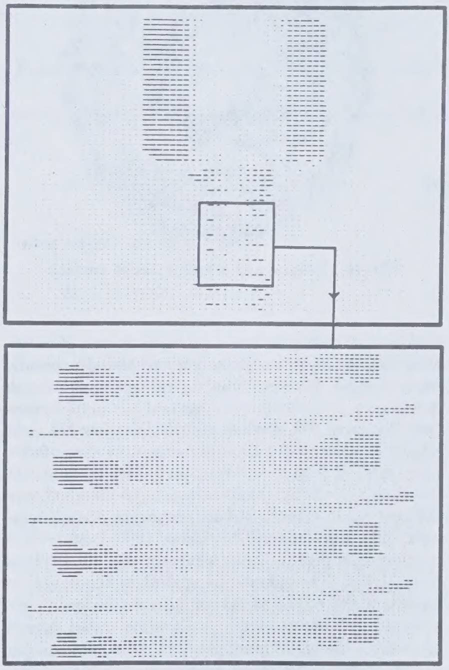
In the photo-sampler implementation described above, all extended sources are positioned in the same plane. As stated earlier, this system is only capable of measuring surface normal vectors that lie in the plane of the sources. To recover the shape and reflectance of general three-dimensional surfaces, a three-dimensional version of the photo-sampler has been developed and described in [Sato et al. 90].



OBJECT



SPECULAR STRENGTH LAMBERTIAN STRENGTH



NEEDLE MAP

Figure 11: Shape and reflectance of a metal bolt recovered using the photo-sampler system. The head of the bolt is painted and has a diffuse surface, while the threaded section is specular.

4 Rough and Textured Surfaces

All surfaces encountered in practice are rough at some level of detail. In many vision applications, surface variations are comparable in dimensions to the resolution of the imaging system. Fig. 12 shows a microscope image of a via-hole filling on a ceramic circuit board. Image intensities produced by such surfaces vary in an unpredictable manner from one sensor element (pixel) to the next. Hence, it is difficult to obtain dense and accurate surface shape information by using existing passive or active sensing techniques, including, the structured highlight and photometric sampling methods described in the previous sections. A practical and reliable solution to this rather difficult extraction problem is desirable.

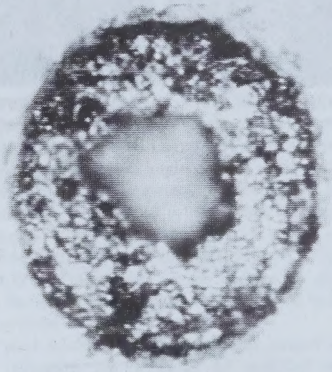


Figure 12: Image of a visibly rough surface.

Focus analysis provides a powerful means of recovering the shapes of visibly rough surfaces. Previously, focus analysis has been used to recover sparse depth maps of general scenes (see [Pentland 87], [Subbarao 89], [Krotkov 87], and [Darell & Wohn 88]). Here, we present a shape-from-focus method [Nayar & Nakagawa 90] for the accurate estimation of visibly rough surfaces. The method uses different focus levels to obtain a sequence of object images. The sum-modified-Laplacian (SML) operator is developed to provide local measures of the quality of image focus. The operator is applied to the image sequence of the object to determine a set of focus measures at each image point. A Gaussian model is used to describe the variation of the SML focus measure due to defocusing. This model is used by a depth estimation algorithm to interpolate a small number of focus measure values and obtain accurate depth estimates. An automated shape from focus system has been implemented using an optical microscope and tested on a variety of industrial samples.

4.1 Shape from Focus

Fig.13 shows an object of unknown shape placed on a translational stage. The reference plane shown corresponds to the initial position of the stage. The configuration of the optics and sensor defines a single plane, the "focused plane," that is perfectly focused onto the sensor plane. The distance d_f between the focused and reference planes, and the displacement d of the stage with respect to the reference plane, are always known by measurement. Consider the surface element, s, that lies on the object surface, S. If the stage is moved towards the focused plane, the image of s will gradually increase in its degree of focus (high frequency content) and will be perfectly focused when s lies on the focused plane. Further movement of the element s will again increase the defocusing of its image. If we observe the image area corresponding to s and record the stage displacement $d = \overline{d}$ at the instant of maximum focus, we can compute the height d_s of s with respect to the stage as $d_s = d_f - \overline{d}$. This procedure may be applied independently to all surface elements to obtain the shape of the entire surface S.

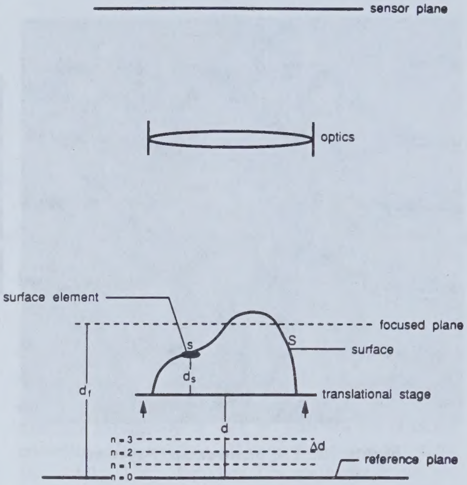


Figure 13: Shape from focus.

To automatically detect the instant of "best" focus, we will develop an image focus measure. In the above discussion, the stage motion and image acquisition were assumed to be continuous processes. In practice, however, it is not feasible to acquire and process such a large number of images in a reasonable amount of time. Therefore, we obtain only a finite number of images; the stage is moved in increments of Δd, and an image is obtained at each stage position. An interpolation algorithm computes accurate depth estimates from a small number of focus measures.

4.1.1 A Focus Measure Operator

In [Nayar & Nakagawa 90] the effects of defocusing are analyzed in frequency domain. Defocusing attenuates the high frequencies in the focused image and hence plays the role of a *low-pass* filter where the bandwidth decreases with increase in defocusing. Therefore, the effects of defocusing are more pronounced and detectable if the image has strong high-frequency content. An effective focus measure operator, therefore, must high-pass filter the image. One way to high-pass filter an image is to determine its second derivative. For two-dimensional images, the *Laplacian* may be used:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{4}$$

where I is the image intensity at the point (x, y). In [Nayar & Nakagawa 90] we have shown that the result of applying the Laplacian operator to an image point may be assumed to be a Gaussian function of the defocusing parameter σ_h. This general behavior is expected irrespective of the focus measure operator used.

In the context of textured images, the Laplacian poses a problem as a focus measure operator. Note that in the case of the Laplacian the second derivatives in the x and y directions can have opposite signs and tend to cancel each other. In the case of textured images, such instances may occur frequently and the Laplacian may at times behave in a unstable manner. We overcome this problem by defining the *modified Laplacian* as:

$$\nabla^2_M I = \left|\frac{\partial^2 I}{\partial x^2}\right| + \left|\frac{\partial^2 I}{\partial y^2}\right| \tag{5}$$

Note that the modified Laplacian is always greater or equal in magnitude to the Laplacian.

The discrete approximation to the Laplacian is usually a 3×3 operator. In order to accommodate for possible variations in the size of texture elements, we compute the partial derivatives by using a variable spacing (*step*) between the pixels used to compute the derivatives. Hence, the discrete approximation to the modified Laplacian is computed as:

$$\begin{aligned} ML(x,y) &= \mid 2I(x,y) - I(x - step, y) - I(x + step, y) \mid \\ &+ \mid 2I(x,y) - I(x, y - step) - I(x, y + step) \mid \end{aligned} \tag{6}$$

Finally, the focus measure at a point (i,j) is computed as the sum of modified Laplacian values, in a small window around (i,j), that are greater than a threshold value:

$$F(i,j) = \sum_{x=i-N}^{i+N} \sum_{y=j-N}^{j+N} ML(x,y) \quad for\ ML(x,y) \geq T_1 \tag{7}$$

where the parameter N determines the window size used to compute the focus measure. In contrast to auto-focusing methods [Krotkov 87], we typically use a small window of size 3×3 or 5×5, i.e. $N = 1$ or $N = 2$. We shall refer to the above focus measure as the *sum-modified-Laplacian* (SML).

4.1.2 Depth Estimation

The focus measure function at any given image point may be denoted by $F(d)$; the SML focus measure as a function of the distance of the corresponding surface point from the focused plane (Fig.13). As stated in the previous section, the peak of $F(d)$ can be modeled using a Gaussian distribution with mean value $\overline{d}$ and standard deviation σ_F (Fig.14). Using the Gaussian model, the focus measure function may be expressed as:

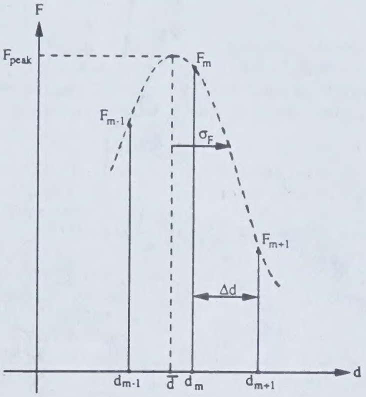


Figure 14: Gaussian interpolation of focus measures.

$$F = F_p \ exp \left\{ -\frac{1}{2}\left(\frac{d - \overline{d}}{\sigma_F}\right)^2 \right\} \tag{8}$$

Using natural logarithm, we obtain:

$$ln\ F = ln\ F_p - \frac{1}{2}\left(\frac{d - \overline{d}}{\sigma_F}\right)^2 \tag{9}$$

The above Gaussian model enables us to compute depth $\overline{d}$ from just three focus measures. The fringes of the focus measure function may deviate from the Gaussian distribution since the magnification of the imaging system can vary substantially from one fringe to the other. We use three focus measures that lie on the largest mode of $F(d)$ (Fig.14).

4.2 Automated Shape from Focus System

We have implemented a fully automated shape from focus system for the recovery of microscopic objects. A photograph of the system is shown in Fig.15. A Nikon Alphaphot-2 model microscope is used to image the objects. Objects can be magnified using objective lenses with ×10, ×40, and ×100 magnification. The object is illuminated using bright field illumination where light energy is focused on the object by the same lenses that are used to magnify the object. A CCD camera with 512×480 pixels is mounted on the microscope to obtain digital images of the object. The z-axis of the microscope stage is driven by a stepper motor and the position of the stage can be computer controlled with a resolution and accuracy of 0.02 μm. The shape from focus algorithm is programmed and executed on a Sun workstation.

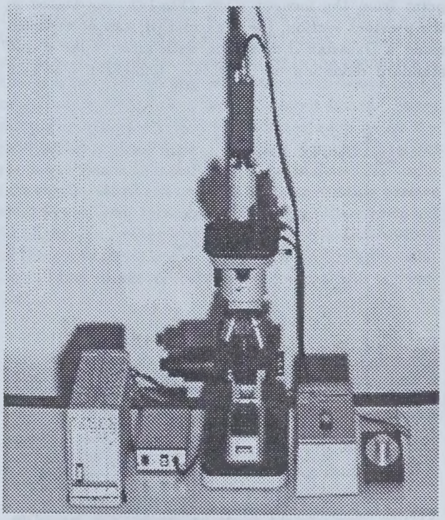


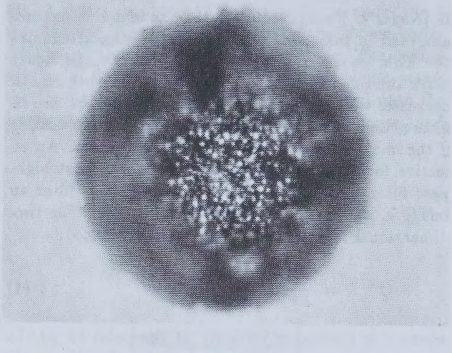
Figure 15: Automated shape from focus system.

The object is placed on the microscope stage and the appropriate objective lens is used to magnify it. The focus measure parameters (T_1 and *step*) and the stage displacement (Δd) are provided by the user. The program then automatically increments the stage position, digitizes and stores an image for each new position, and uses the image sequence to compute a depth map of the object. Fig.16 shows results obtained for a tungsten paste filling in a via-hole on a ceramic substrate. The via-hole is approximately 70 μm in diameter. A total of 18 images of the via-hole were obtained using stage position increments of $4\mu m$. Several other results are reported in [Nayar & Nakagawa 90].

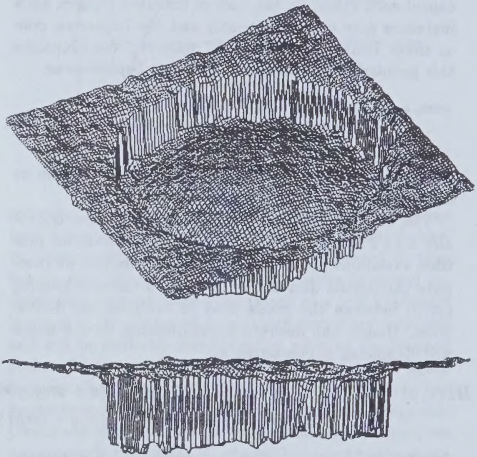
5 Conclusion

In this paper, we have presented new shape recovery methods for three-dimensional machine vision. We conclude with the following remarks:

- The results presented here demonstrate that it is possible to develop effective shape recovery techniques but focusing on a particular class of surfaces. The three classes we have examined here are: specular surfaces; surfaces with varying reflectance; and rough and textured surfaces. Each class encompasses a wide range of real-world objects and inspection problems.
- The shape recovery methods developed here are: structured highlight; photometric sampling; and



Camera image.



Depth maps

Figure 16: Depth maps of a via-hole filling computed using the shape from focus system.

shape from focus. These methods are based on physical models of surface reflectance and image formation. In general, the estimation of three-dimensional shape from a single two-dimensional image is an under-constrained problem. We have used active illumination and sensing techniques to constrain the problem and recover accurate shape estimates.

- Three automated vision systems have been developed that are based on the above recovery techniques: the SHINY system for specular surfaces; the photo-sampler system for surfaces with varying reflectance; and the shape from focus system for rough and textured surfaces. Each system has been tested on industrial samples. The results demonstrate that these systems can provide practical solutions to a variety of inspection problems.

Acknowledgements

The systems presented in this paper were developed in collaboration with Arthur Sanderson, Lee Weiss, and David Simon (SHINY system); Katsushi Ikeuchi and Takeo Kanade (Photo-sampler system); and Yasuo Nakagawa and Ushir Shah (shape from focus system). This research was conducted at the Robotics Institute, Carnegie Mellon University; the Production Engineering Research Lab., Hitachi Ltd.; and the Center for Research on Intelligent Systems, Columbia University.

References

[Coleman & Jain 82] E. N. Coleman and R. Jain, "Obtaining 3-dimensional shape of textured and specular surfaces using four-source photometry," *Computer Graphics and Image Processing*, Vol. 18, No. 4, pp. 309-328, April, 1982.

[Darell & Wohn 88] T. Darrell and K. Wohn, "Pyramid Based Depth from Focus," *Proc. Computer Vision and Pattern Recognition Conf.*, pp. 504-509, 1988.

[Horn 70] B. K. P. Horn, "Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," MIT Project MAC Internal Report TR-79 and MIT AI Laboratory Technical Report 232, November, 1970.

[Horn 84] B. K. P. Horn, "Extended Gaussian Images," *Proceedings of IEEE*, Vol. 72, No. 7, December 1984.

[Ikeuchi 81] K. Ikeuchi, "Determining surface orientations of specular surfaces by using the photometric stereo method," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 3, No. 6, pp. 661-669, November, 1981.

[Krotkov 87] E. Krotkov, "Focusing," *International Journal of Computer Vision*, Vol. 1, pp. 223-237, 1987.

[Nayar et al. 90] S. K. Nayar, A. C. Sanderson, L. E. Weiss, D. D. Simon, "Specular Surface Inspection Using Structured Highlight and Gaussian Images," *IEEE Tran. on Robotics and Automation*, Vol. 6, No. 2, pp. 208-218, April, 1990.

[Nayar & Nakagawa 90] S. K. Nayar and Y. Nakagawa, "Shape from Focus: An Effective Approach for Rough Surfaces," *Proc. IEEE Robotics and Automation Conf.*, pp. 218-225, May 1990.

[Nayar, Ikeuchi, & Kanade 90]
S. K. Nayar, K. Ikeuchi, T. Kanade, "Determining Shape and Reflectance of Hybrid Surfaces by Photometric Sampling," *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 4, pp. 418-431, August, 1990.

[Nayar 90] S. K. Nayar, *Shape Recovery using Physical Models of Reflection and Interreflection*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Carnegie Mellon University, December, 1990.

[Nayar, Ikeuchi, & Kanade 91]
S. K. Nayar, K. Ikeuchi, T. Kanade, "Surface Reflection: Physical and Geometrical Perspectives," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 7, pp. 611-634, July 1991.

[Pentland 84] A. P. Pentland, "Local Shading Analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 6, No. 2, pp. 170-187, March, 1984.

[Pentland 87] A. Pentland, "A New Sense for Depth of Field," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 9, No. 4, pp. 523-531, July 1987.

[Sanderson, Weiss, & Nayar 88] A. C. Sanderson, L. E. Weiss, and S. K. Nayar, "Structured Highlight Inspection of Specular Surfaces," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 1, pp. 44-55, January, 1988.

[Sato et al. 90] H. Sato, S. K. Nayar, and K. Ikeuchi, "Implementation and Evaluation of a Three-Dimensional Photometric Sampler," Technical Report, Carnegie Mellon University, November 1990.

[Subbarao 89] M. Subbarao, "Efficient Depth Recovery through Inverse Optics," *Machine Vision for Inspection and Measurement*, edited by H. Freeman, Academic Press, 1989.

[Woodham 78] R. J. Woodham, "Photometric stereo: A reflectance map technique for determining surface orientation from image intensity," *Proc. SPIE*, Vol. 155, pp. 136-143, 1978.

Sensor Planning for Reactive Robotic Systems

Gregory D. Hager
Department of Computer Science
P.O. Box 2158 Yale Station
Yale University
New Haven, CT 06520

Gerhard Grunwald
German Aerospace Research Establishment
Institute of Robotics and Systems Dynamics
Münchenerstr. 20
D-8031 Oberpfaffenhofen

Abstract

Recently, there has been increased emphasis on employing reactive actions in robot task planning. The principle reasons for this change are to increase the robustness of robot actions by making them sensor controlled, and to accommodate dynamic, unpredictable environments. However, in many cases, supporting reactive mechanisms requires choosing task level sensor inputs for the reactive procedure. This paper addresses the issue of planning the task level sensing required to carry out a reactive robot program. A preliminary framework for planning is presented, and sensor planning is illustrated for the problem of replacing a mechanically attached plug in a space environment.

1 Introduction

Reactive robotics is an umbrella term for the philosophy that robotic tasks should be carried out using direct, task level sensor feedback. The efficacy of using task level sensor feedback is indisputable: task level feedback increases both the range and reliability of robot task performance. Unfortunately, the vagaries and complexities of processing data from sensors capable of task level sensing (most notably vision, dense and sparse range sensors, tactile sensors, and force sensors) has limited progress in this area. Thus far, most interesting systems employ rudimentary, "hard-wired" sensor inputs to accomplish simple, unstructured tasks such as wandering through corridors or collecting rocks on the surface of Mars [2, 6].

More recently, there have been attempts to employ reactive procedures in more structured tasks by posing task level planning as the configuration of generic reactive modules to accomplish specific objectives [5, 13, 14]. In principle, planning for reactive behavior can be decomposed into the following two subproblems:

- Analysis of the task and its decomposition into sensor-relative actions. This is, in many ways, similar to "classical" planning, however, it is important to deal with problems of realtime operation and concurrency [14].
- An analysis of the structures in the world that can be recognized and used to guide actions. The goal to reduce the required sensor information to the minimal, most reliable sensor inputs needed to perform the action.

We refer to the former as *task level planning* and the latter as *sensor level planning.* Intuitively, task level planning is concerned with the operational decomposition of the task into more basic actions, and sensor level planning is concerned with satisfying the information needs of these actions. Although we have separated these two activities, there are clearly a number of interactions between these two modules that need to be understood.

In this paper, we further define the problem of sensor level planning, we propose a general procedure for accomplishing it, and we illustrate the planning algorithm on a benchmark problem. We assume an environment structured to the point that the sensor planning system can use simulation as a reliable means of testing and debugging sensor plans. We note that the amount of structure needed by sensor planning is still somewhat unclear. It appears to depend heavily on the type of actions that must be planned, as well as the sensor information required to support them. In our case, we are mostly interested in sensor planning to support space teleoperation. Thus, we attempt to infer or elicit sensor-based actions from user input, and build the sensing configuration required to support it. In general, space environments are well-structured, and as described below, it is often possible to build simulations of high-fidelity.

The remainder of this article is structured as follows. In the next section, we explore the basic structure of the planning problem and introduce our test environment and example problem. In Section 3, we describe the structure of the planning system and discuss how it would plan a sensor-guided motion. Finally, we conclude with a discussion of our current work.

2 Task Level Planning and Sensor Level Planning

For the purposes of this paper, we will refer to actions that are executed without any task level sensing as *open-loop*, and actions using sensing as *closed-loop*. The role of task level planning is to construct a robot program composed of primitive, open-loop and closed-loop actions that successfully achieves some objective. Henceforth, we will assume that the task level planner is able to recognize that some actions cannot be executed open-loop with enough precision to succeed, and that it can infer task-specific geometric goal conditions that must be satisfied in object relative coordinates. Any objects referenced by a task description are known to the system *a priori* and geometric models are available. Furthermore, depending on the structuring of the environment, the approximate position or pose of object may be known. Finally we assume that the task level planner has enough geometric information to decompose closed-loop actions into approximately straight line motions. That is, if moving from position A to position B involves moving around an intermediate object, then the planner should define intermediate positions (or sets of positions) so that the total motion is a series of approximately straight segments. This type of decomposition can be done using relatively coarse, nonparametric geometry such as the maps described by Elfes [4].

As stated above, the goal is to rely on as little calibration as possible, and to emphasize object-relative positions rather than robot-relative positions. So, for example, a robot motion to some point A in the workspace involves describing the position A *relative to some features of the environment* rather than in a global robot coordinate system. Hence, closed-loop actions lead planning away from a classical "robot-centered" definition of motions and toward one that is *object-centered.* This is similar in conception to the indexical-functional representation proposed by Agre and Chapman [1].

In order to carry out sensor-based actions, a robotics system must recognize and/or verify the existence and pose of the "landmarks" used to define relative positions, locate specific features that can be tracked to control the manipulator, perform the action, and at the same time monitor for sensor inputs that indicate the desired goal position has been reached. We initially classify the type of sensor operations needed in each of these steps as a *static* sensor task or a *dynamic* sensor task. Static sensor tasks perform one-time measurements and calculations. These actions are purely informatory, and are usually added to a program to instantiate variables needed by later actions. For example, recognizing an object addresses the problem of matching sensor information with a model. This information can be used to relativize an action to a particular object, to verify expected initial conditions, or to aid in the search for specific object features needed to control an action. The sensor planning problem is to match the available sensor information with a matching or recognition procedure. Dynamic sensor tasks support control of *performatory actions*: actions that lead to actual motion of the robot. For example, approaching an object requires tracking lines or patterns that can control the attitude and relative position of the robot with respect to the goal position. The sensor planning problem is to choose the type of sensing and the sensor patterns to be used in the feedback loop. It must ensure that the entire action, from initial state to final state can be carried out using some combination of sensing, inference, and servoing.

The connection between actions proposed by the task level planner and the operations planned at the sensor level takes two forms. For informatory actions, the action generated by the task planner corresponds directly to a task to be planned by the sensor planner. In some cases, the task level planner is allowed to specify accuracy requirements or other conditions that must be met. For example, a pose estimate may need to meet accuracy requirements in order to support a particular fine motion strategy. Performatory actions that require sensing support are "decomposed" into sensing tasks by the sensor planner. This decomposition is governed by the contextual constraints surrounding the action. For example, approaching an object requires recognition, pose estimation, and tracking to support control. The sensor planner also has information on the type of control that is taking place so that it can choose the features to be tracked based on control stability criteria. We discuss these issues in greater depth in the next section.

2.1 Experimental Planning Environment and Example

Our experimental environment consists of the ROTEX (RObot Technology EXperiment) robot, sensors and workcell. ROTEX is a German space telerobotic experiment that will fly with the next German space lab mission (D2) in 1993. It consists of a six-axis robot equipped with a sensorized gripper and an external fixed pair of video cameras providing a stereo image of the robot workspace. The gripper is provided with a number of sensors: two six axis force-torque wrist sensors, two tactile arrays, grasping force control, an array of 9 laser-range finders and a tiny pair of stereo cameras. ROTEX is designed to operate autonomously, to be teleoperated by astronauts, or to be teleoperated from a ground station. A more detailed description of the ROTEX environment can be found in [9] or [10]. Although the ROTEX environment was primarily designed for telemanipulation, there are additional interfaces in the laboratory version of ROTEX that support other modes of operation. We note that, for the moment, the configuration used for sensor planning experiments only uses a single hand-camera, the hand-mounted range sensors, and a single global-camera: no stereo information is available.

Our sensor planning framework will be illustrated using one of the D2 mission experiments: the task of finding, grasping and removing the *ORU* (Orbit Replaceable Unit), an

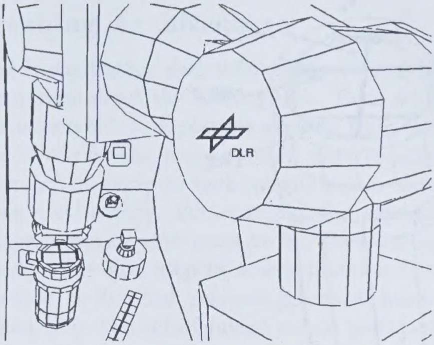


Figure 1: The ROTEX workcell.

electro mechanical plug (See figure 2). The *ORU* is removed by expanding a grip on the top, executing the proper twisting motion, and then pulling away from the attachment surface. We will consider planning for the recognition and sensor-guided approach to the ORU.

We note that experiments with the system in teleoperated mode have shown that this task is extremely difficult to carry out with no sensor feedback. The tolerances on the hand position and orientation are very small, and the sensor information available to the human operator is often difficult to interpret. The system permits the operator to use the four laser distance sensors pointing out of the fingertips to control gripper orientation near the *ORU*. This simplifies the task greatly. We have recently developed a vision-based controller that can control gripper position parallel to the surface of the *ORU*. Combining both range and vision leaves the operator the simple task of getting the robot near the *ORU*, turning on sensor-based control, and then controlling the z axis motion until the gripping position is reached.

3 The Sensor Models and Planning Architecture

As noted in the previous section, requests to the sensor planner are either basic sensor requests or higher level actions that are decomposed by the sensor planner into a sequence of more basic sensing actions. For example, the high level action *approach(ORU,final_pos)* is decomposed into the following operations: *approach(ORU,final_pos)* :=[*recognize(·); pose(·); tracking(·)*]. The first two operations are static, informatory actions, while the latter is used to support performance of a dynamic closed-loop motion. The sensor planner constructs a plan for these actions using information available on the *ORU*, the available sensors, and final position criteria supplied in the request.

Information on objects and sensors comes from object models, sensor models, and sensor simulations. The object models are from the CATIA CAD modeler which provides us with a boundary representation from which physical edges and planar surfaces can be

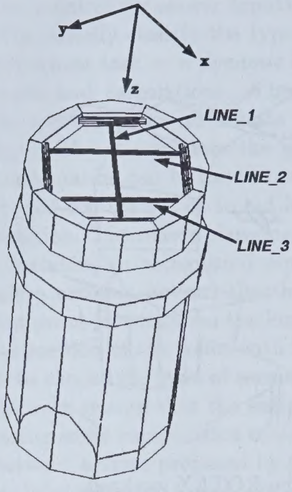


Figure 2: The Orbit Replaceable Unit.

easily extracted. In addition, we separately represent artificial markings such as the guide lines on the ORU. Using this information, we have implemented and tested simulations for the distance sensors, a laser range sensor, and a 6 dof force/torque sensor. Further we have also implemented a primitive camera simulation which is currently being extended to deal with more sophisticated features and imaging models.

3.1 Logical Sensors

The basic representation in the system is a structure similar to the notion of Logical Sensors [8]. The basic form of the structure is:

Logical Sensor	<name>
Input_List	{<Logical Sensors>}
Data_Proc	{<return> = <data_proc(...)>}
Type	**static | dynamic**
Controller	{<ctrl>}
Valid	{<cond>}
Characteristic	<pose>; <fixed|movable>; <field of view>; <...>

At planning time, these logical sensors are instantiated and organized into a network structure based on various choices of input sources from the *Input_List* and data processing procedures from the *Data_Proc* list. Each input process has a return value which is used as input to a data processing algorithm. The *Type* slot indicates whether this sensor provides information just once, or over an interval of time. The *Valid* slot represents constraints that must be satisfied by the input to the chosen data procedures in order for them to function correctly and reliably. The *Controller* must check the sensor data with respect to *Valid* and, in case of a dynamic operation, coordinate and synchronize the instantiated data processes. If the Logical Sensor represents a physical sensor it may also control the device. Physical sensors use the *Characteristic* slot to represent information like the position, the field of view, work-criterions and very important if its a fixed sensor or its mounted on a movable device.

3.2 Sensor Planning Architecture

Planning is viewed as instantiating and refining the logical sensor associated with an action based on information about the sensing task. Each action of this type has three parameters: the information the task planner expects from the sensor system; the subject; and, the goal conditions for the entire request. Goal conditions on sequence requests like *approach* are extended to apply to each atomic sensor request and form a common condition the sequence must satisfy. For example, the *approach* action supplies a final position and positioning accuracy that must be attained during the course of executing the three basic actions. The first step in sensor planning is to perform a geometrical analysis of the *ORU* model to find task relevant geometric information that is observable by the available sensors from the set of initial sensor positions. For example, the *recognize()* request requires discriminating geometric features which uniquely identify the *ORU*. To this point, the features used in our environment are defined by hand and include lines, faces, special surface markings and combinations thereof. The geometrical analysis results in a set of geometric classes which are detectable by the sensors. In case of *ORU* the geometric classes are line, face and the relation of the three markings on the top.

The next step is to build a tree of logical sensor which can observe the object features, and to ensure that the execution of the associated data processing routines will be successful. The root of the tree is the logical sensor corresponding to the sensing task. Tree growth is constrained by knowledge of the feature classes computed in the previous geometrical analysis step. Starting from the root for each geometric entity all suitable pairs of Logical Sensors of the Input_List and data processes of Data_Proc define successor nodes. This process is continued recursively until the level of physical sensors is reached.

The class tree is then pruned and instantiated by using task information and output from the sensor simulation system based on the configuration expected when the plan will be executed. For each physical sensor a simulated measurement is performed and the observable geometries are evaluated with respect to the different data processes. At each node the validity conditions are tested, and if a test fails the node is pruned. So the result of the instantiation is a tree containing only reliable information. In our example, the camera simulation will find all lines and faces seen in the Figure 2. The filtered geometry is based on the sensor observation model and hence represents only the visible parts of it.

The instantiated tree is then analyzed with respect to the goal criteria for completeness and redundancy. Completeness addresses the problem if the expected sensor information is sufficient to attain required accuracy, stability, or uniqueness conditions. In case of *recognize* the simulated information is checked if it uniquely identifies the *ORU*. Redundancy is an important consideration as it increases the robustness of sensor-based action, but it also increases the complexity of data processing. In some case, the sensor planner may need to choose a compromise between safety and performance of the system. The failure of any of the above steps causes the sensor planner to reject the request as unsatisfiable.

3.3 Constructing Plans for Reactive Actions

Just as *recognize(·)* has a corresponding logical sensor, so do dynamic actions such as tracking. We view the problem of planning sensing to move between two points as the problem of constructing the observer for a series of *adaptive regulators* that attain the final desired position. These regulators are "constructed" by configuring pre-defined generic regulators using the appropriate choice of sensor inputs and goal patterns. More precisely, we are considering the following problem.

Given:

- a geometric object description;
- a model of sensor feature detection;
- a range of initial positions defined relative to the object; and
- a goal position defined relative to the object.

Determine:

- a collection of "regulators" and sensor inputs;
- the range of stable regulation for each; and
- a switching behavior.

So That:

- All plausible paths from the initial state to the goal position are covered by a stable regulator.
- The system deviates as little as possible from the straight line path from the initial state to the goal state.

We note that, although there is a great deal of discussion of the local and global stability of regulation systems [15], we do not know of any computational methods for synthesizing a regulator and analyzing its region of stability. Moreover, while this synthesis problem could, in principle, be formulated analytically, we postulate that any completely analytic solution would be computationally infeasible to carry out within the time constraints imposed by practical planning.

Consequently, we have begun to explore approximate solutions that involve decomposition of general motions into combinations of specific motions for which the regulation problem is well understood. Specifically, we note that if a goal pattern is in view of the camera, then servoing forward while holding a point on the pattern in the center of the image will lead the system to translate to the point regardless of orientation. Consequently, one way of positioning the gripper is to approach a pattern until a distance is reached where other visual patterns or visual and range information can be used to control orientation as well as position. Furthermore, the gripper can rotate about an object by "linking" together visual patterns that contain enough information to control distance (via scale) as well gripper/object relative orientation. Thus, an effective motion plan consists of: 1) finding a pattern on the same surface as the goal that is visible; 2) proceeding toward that pattern; 3) rotating or navigating about the surface until the goal is visible; 4) orienting and translating to the goal position.

For the gripper-mounted camera, the available information sources are various types of gradient-based pattern tracking as well as feature-based tracking. As the camera moves, these patterns distort and scale. It is relatively straightforward to determine the range of visual angles and scales over which particular features or patterns are detectable [16]. In general, stability degrades as the camera moves away from objects.[1] This leads us to consider a representation of the stable region of control in terms of sections of a sphere. Recalling that stability at a point can be determined by examining the Hessian of the objective function describing the controller [12, 15], we can find the boundaries of the region of stable control by testing points on the surfaces of a spheres.

This basic idea leads us to consider a sensor planning algorithm using methods similar to the backprojection technique employed by Donald's motion planner [3, 11]. Essentially, the gripper is simulated at the goal position and a stable set of sensor inputs is chosen. The stable point is expanded to a stable region. Regulators are chosen on the boundary of this region and extended further until the set of goal configurations is encompassed.

More formally, if s is a logical sensor, let D_s denote the spherical section (essentially two intervals of angles and one interval of distance) from the goal over which the sensor detects the chosen features. If a regulator r relies on sensors A_r let $\mathcal{D}_r = \cap_{s \in A} D_s$. Hence, $\mathcal{D}_r$ is the range over which r will receive the measurements that it expects. Let $\mathcal{S}_r$ denote the region of stability of r. Then $\mathcal{O}_r := \mathcal{S}_r \cap \mathcal{D}_r$ represents the operating range of the regulator r. If I is a set of initial positions expressed relative to the goal point g, then let $\mathcal{I}$ denote the minimal spherical section such that $\mathcal{I} \supseteq I$. Let $A(\cdot)$ denote the range of angles of a spherical section. Our proposed algorithm then proceeds as follows:

1. Given a goal position g, synthesize a regulator r_g such that r_g is stable at g. Compute $\mathcal{O}_{r_g}$. If $\mathcal{O}_{r_g} \supseteq \mathcal{I}$ then stop.

2. Let L be the upper bound on diameter in $\mathcal{O}_{r_g}$. Synthesize regulators on orientation and distance $r_1, r_2, \ldots r_n$ such that $A(\cup_{i=1}^{n} \mathcal{O}_{r_i}) \supseteq A(\mathcal{I})$ and $L \in \cap_{i=1}^{n} \mathcal{O}_{r_i}$.

3. Create a queue Q of these regulators ordered by increasing maximum stable distance.

4. Iteratively:

    - Remove the first regulator r with operating range $\mathcal{O}_r$ from the queue.
    - Compute a set of regulators $r_1, r_2, \ldots r_n$ on horizontal and vertical translation such that $A(\cup_{i=1}^{n} \mathcal{O}_{r_i}) \supseteq A(\mathcal{O}_r)$. Add them to the queue and sort it. If no such set exists, exit with *failure*.
    - Exit when the first element in the queue has an operating range with a maximal distance component larger the largest distance in $\mathcal{I}$.

Once a collection of regulators is found and the algorithm exits, the corresponding logical sensing network of feature trackers is instantiated and information for their initialization is planned. The final result is a tracking "program" that provides information for stable regulation during the approach operation.

---

[1] We note that perspective projection causes scale ambiguity in camera images so that camera control must be implemented using adaptive methods.

4 Discussion

The basic elements of the system we have described is currently being implemented within the ROTEX environment of Section 2. As noted above, many of the sensor simulations exists, as well as an adaptive vision-based regulator and a range-based regulator. We have a primitive camera simulation which we are planning to augment with a detection model similar to that described in [16]. The point stability analysis has been implemented for the monocular camera and range sensor combination. We are currently extending that analysis to spherical regions as described in Section 3.

Our analysis has been based on a reasonably known, structured environment. In particular, we are able to use the available environment models to constrain the initial conditions for most sensor planning operations. This facilitates efficient offline sensor planning. It appears that working in an unstructured environment requires much more online evaluation of execution conditions as well as a more general framework for task planning. In future work, we hope to consider the issues involved in working in unstructured environments.

Acknowledgements: This research was supported by DARPA grant N00014-91-J-1577, by National Science Foundation grants IRI-9109116 and DDM-9112458, by NATO Collaborative Research Grant CRG-910994, and by funds provided by Yale University.

References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proc. of AAAI-87*, pages 268–272. Seattle, WA, 1987.

[2] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[3] B. R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, 31(3):295–353, 1987.

[4] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[5] J. Firby. An investigation into reactive planning in complex domains. In *Proc. of AAAI-87*, pages 202–206. Seattle, WA, 1987.

[6] E. Gat and D. Miller. Modular, low-computation robot control for object acquisition and retrieval. Jet Propulsion Lab Technical Report, 1990.

[7] C. Hansen and T. Henderson. CAGD-based computer vision. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(11), 1989.

[8] T. Henderson and E. Shilcrat. Logical sensor systems. *Journal of Robotics Systems*, 1(2):169–193, 1984.

[9] G. Hirzinger. The telerobotic concepts of ROTEX—Germany's first step into space robotics. *39th I.A.F., Bangalore, India*, 1988.

[10] G. Hirzinger, G. Grunwald, B. Brunner, and H. Heindl. A sensor-based telerobotic system for the space robot experiment ROTEX. *2. International Symposium on Experimental Robotics*, 1991. Toulouse, France.

[11] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

[12] D. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, New York, N.Y., second edition, 1984.

[13] D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Journal on Robotics and Automation*, 5(3):280–293, June 1989.

[14] D. McDermott. Planning reactive behavior: A progress report. In K. Sycara, editor, *Innovative Approaches to Planning, Scheduling and Control*, pages 450–458. Kaufmann, San Mateo, CA, 1990.

[15] K. Narendra and A. M. Annaswamy. *Stable Adaptive Systems.* Prentice Hall, Englewood Cliffs, N.J., 1989.

[16] K. Tarabanis, R. Tsai, and P. Allen. Automated sensor planning for robotic vision tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 76–82. 1991.

Multidimensional (a, b)-trees: A Close Look at the Neighbor-Support Paradigm*

Vijay K. Vaishnavi Richard R. Mixon
Department of Computer Information Systems
Georgia State University
Atlanta, Georgia 30303, U.S.A.

ABSTRACT: The *neighbor-support paradigm* for the generalization of the balancing concept of a 1-dimensional tree structure to higher dimensions is presented in a general form and its usefulness is demonstrated in a concrete manner. The paradigm is used to define and investigate *multidimensional (a, b)-trees*, a new dynamic data structure that can form the basis of file structures for very large databases to support multi-key or associative queries. The simplicity of the update algorithms for *k*-dimensional (a, b)-trees is demonstrated. It is shown that the data structure retains important properties related to its amortized update complexity, which translate to similar properties for *self-organizing (a, b)-trees*, a new file structure based on 2-dimensional (a, b)-trees.

1. Background

A linearly ordered list of *n* 1-dimensional data items can be organized into a search tree [13]. *Balanced search trees* (e.g., AVL-trees [1], (a, b)-trees [18], and weight-balanced trees [18]) have the desirable property of requiring O(log *n*) time for access, insertion, deletion, and certain other operations. A *k-dimensional search tree* [4, 12] is a natural dynamic data structure for linearly ordered *k*-dimensional data. Such a structure, however, needs to be appropriately balanced to ensure efficient worst-case performance (see [22] for an alternative approach). A *balanced k-dimensional search tree* is an appropriately balanced *k*-dimensional search tree that can support access as well as update operations in O(log *n* + *k*) time, which is optimal [11].

A balanced (*k* + 1)-dimensional tree structure can be used to define an efficient *dynamic weighted k-dimensional tree structure*, by letting the lowest level subtrees be used as virtual subtrees representing the weights of the (*k*-dimensional) data items (see [9]). When the weights are interpreted as access probabilities, a dynamic weighted *k*-dimensional tree is called a *self-organizing k-dimensional tree*; it dynamically adapts itself to the access pattern which may be non-uniform. The best that can be expected from a self-organizing *k*-dimensional tree is that its access time is "logarithmic," i.e., $O(\log W/w_i + k)$, where w_i is the access probability of item *i* at a certain time and *W* is the sum of all the access probabilities at that time (see [18]).

Interest in developing and investigating balanced *k*-dimensional trees is on-going (see, for example, [3, 5, 6, 7, 8, 15, 16, 17, 20, 21, 25, 26, 28, 29]). The research objective of the authors is to develop and investigate "optimally-balanced" multidimensional, weighted, and self-organizing tree structures through "optimal" generalizations of well-known balanced 1-dimensional tree structures [e.g., 1, 2, 10, 23]. The resulting tree structures must have simple and efficient update algorithms and must retain important properties of the corresponding base structures. Such tree structures are expected to play a useful role in the design of efficient and simple data structures for *k*-dimensional data (see [14, 15, 19]). Vaishnavi has developed

---

*This work is partially supported by a grant from the Research Committee of the College of Business Administration at Georgia State University.

a number of data structures that meet the above objectives [25, 26, 28]. Multidimensional AVL-trees [25] generalize the balancing concept of AVL-trees [1] to higher dimensions. Weighted leaf AVL-trees [26, 27] are based on 2-dimensional leaf AVL-trees. Multidimensional balanced binary trees [28, 29] generalize the balancing concept in balanced binary trees [23] (same as symmetric binary B-trees [2] or red-black trees [10]). Like AVL-trees, multidimensional AVL-trees and weighted leaf AVL-trees require the application of a constant number of restructuring operations (rotations). k-dimensional balanced binary trees require $O(k)$ rotations for insertion and deletion. The amortized rebalancing time for self-organizing balanced binary trees is constant [29]. Interestingly, the flexibility in the update of these tree structures is not at the expense of their access time (see [26, 29, 30]).

The data structures proposed in [25, 26, 28, 29] follow a paradigm for the generalization of the balancing concept of a 1-dimensional tree structure to higher dimensions. The paradigm is called the *neighbor-support paradigm*. (See [12] for a comparison with the *local-support* and *global-support* paradigms.) The height-analysis of such tree structures is rather difficult, but a recently developed approach [30] seems to have general applicability.

The purpose of this paper is to present the neighbor-support paradigm in a general form and to show its usefulness in a concrete way. The paradigm is used to define *multidimensional (a, b)-trees*, a new data structure of independent interest. The height of k-dimensional (a, b)-trees is analyzed and the simplicity of the update and deletion algorithms is demonstrated. It is shown that the data structure retains important properties related to its amortized update complexity, which translate to similar properties for *self-organizing (a, b)-trees*, a new data structure based on 2-dimensional (a, b)-trees.

2. The Neighbor-Support Paradigm and Multidimensional (a, b)-trees, $b \geq 2a - 1$

A *k-dimensional search tree* is either empty (represented by an external node) or is defined as follows. The k-dimensional keys (attributes) of each k-dimensional vector, x_k in $(x_k, x_{k-1}, \ldots, x_1)$, are organized into a search tree, a k-dimensional *component* of the entire tree. In addition, each k-dimensional key (data item) in a node, a *k-dimensional (internal) node*, points through an *EQSON-pointer* [4] to an *EQSON (k - 1)-dimensional search tree* for the $(k - 1)$-dimensional projection data obtained by omitting x_k, the identical k-dimensional key of the given set of vectors; the EQSON subtree is empty if and only if $k = 1$. The *size* of the k-dimensional tree is the number of 1-dimensional keys stored in the tree. Figure 1.a shows a 2-dimensional ternary search tree (omitting the empty subtrees) of size 10 that stores 2-dimensional vectors.

Observe that each node of a k-dimensional tree other than its root has zero, one, or two *neighboring* EQSON subtrees (see Figure 1.a); the root of the tree as well as the roots of EQSON subtrees do not have any such neighboring EQSON subtrees, each of the nodes on the leftmost and rightmost paths of the tree have only one such neighboring EQSON subtree, and any other node has two such neighboring EQSON subtrees. A node in an i-dimensional component of the tree can only have an $(i - 1)$-dimensional component as its neighbor; all the zero-dimensional components are, of course, empty. In Figure 1.a, the node storing 'a' and 'b' has the 1-dimensional component storing '1', '2', and '3' as its right neighbor; the latter is also a left neighbor of the node storing 'd'.

The *neighbor-support paradigm* for generalizing the balancing concept of a 1-dimensional tree structure to higher dimensions specifies that the "generalized" k-dimensional tree must follow the balancing concept of the 1-dimensional structure (at nodes other than the root of the tree and the roots of the EQSON subtrees) except that there can be a "structure violation" at a node if it is "supported" by at least one of the corresponding neighboring EQSON subtrees. (Cf. the global-support strategy which specifies that the structure violation "must be supported by each of the *adjoining* subtrees of the node causing the structure violation" [12].)

Let us apply the paradigm to the generalization of an (a, b)-tree to higher dimensions. A structure violation at a node N in a *neighbor-supported balanced multidimensional tree* is said to be *directly supported* if it is "supported" by a sibling EQSON subtree of N; otherwise, the structure violation is said to be *indirectly supported*. If a node N in a k-dimensional tree has two neighboring EQSON subtrees then at least one of them is a sibling EQSON subtree. If the node has only one neighboring EQSON subtree then it must be a sibling EQSON subtree. In Figure 1.b, a node in chain A has only one neighboring EQSON subtree and it is a sibling EQSON subtree; a node in chain C has two sibling EQSON subtrees; a node in chain D has one sibling EQSON subtree and one non-sibling (but neighboring) EQSON subtree.

As a structure violation, we will let a k-dimensional (a, b)-tree have a node N at height h which stores less than a - 1 keys (a *shallow-node structure violation*) including zero keys, i.e., a missing node at height h (a *height-gap structure violation*) provided that it is "supported" by N having a neighboring EQSON subtree of height h. Figure 1.a is actually a 2-dimensional (2, 3)-tree. Figure 1.b is another form of this tree in which non-existing (underfilled) nodes at certain heights (height-gap structure violations) are shown by unary nodes. Observe that the structure violations at nodes in chains A and C are supported by the EQSON subtree rooted at B; the structure violation at nodes in chains D and F are supported by the EQSON subtree rooted at E; the structure violation at H is supported by the EQSON subtrees rooted at G as well as I; the structure violation at J is supported only by the EQSON subtree rooted at I.

3. Height-Analysis

The height-analysis of neighbor-supported balanced multidimensional trees is rather difficult [25, 26] precisely because structure violations may be indirectly supported. Vaishnavi and Korson [30] have, however, recently introduced a new approach for height-analysis of k-dimensional leaf and node AVL-trees which seems to have general applicability to neighbor-supported balanced multidimensional trees (see [29]). In this approach it is shown through a transformation procedure that for every tree T_1 of height H and size n, in a neighbor-supported class of trees, B, there is a tree T_2 in a "locally-supported" subset L of B (in which all structure violations in trees are directly supported) whose height is H and size is at most $2^{k-1} \cdot n$. The transformation is applied successively to the subtrees rooted at the internal nodes of a given tree T_1 in order of decreasing dimensions of their nodes and within nodes of the same dimension in decreasing order of their heights. The result of the transformation is summarized in the following result.

Lemma 3.1: Corresponding to a k-dimensional (a, b)-tree of size n and height H, there is a k-dimensional "locally-supported" (a, b)-tree (in which all structure violations are directly supported) of height H and size at most $2^{k-1} \cdot n$.

An upper bound for the height of a k-dimensional (a, b)-tree is now found using the above result and the upper bound on the height of a k-dimensional locally-supported (a, b)-tree, as follows. Let $n(h, k)$ be the minimum size of a k-dimensional locally-supported (a, b)-tree of height h.

Lemma 3.2: For $i \geq 2$, $n(h, i) \geq n(h - 1, i - 1)$

Proof: Straightfoward (by induction). ■

Lemma 3.3: $n(h, k) = n(h - k + 1, 1)$.
Proof: Follows from Lemma 3.2. ■

Lemma 3.4: The height of a k-dimensional locally-supported (a, b)-tree of size n is at most $\log_a ((n + 1)/2) + k$.
Proof: Follows from Lemma 3.3 (see [18]). ■

Theorem 3.5: The height of a k-dimensional (a, b)-tree of size n is at most $\log_a ((n + 1)/2) + ((k - 1)/\log_2 a) + k$.
Proof: Follows from Lemmas 3.1 and 3.4. ■

Corollary 3.5.1: The height of a k-dimensional (2, 4)-tree of size n is at most $\log_2 ((n + 1)/2) + 2k - 1$. ■

4. Insertion and Deletion Algorithms

The main advantage of neighbor-supported balanced multidimensional trees is that the restructuring operations used in the generalized tree are almost the same as those used in the corresponding base structure. If the given tree is empty then the insertion of a k-dimensional vector results in a chain of EQSON-nodes. If the tree is not empty the search will end at an i-dimensional internal node P, not containing x_i, with an empty subtree indicating the absence of x_i in the i-dimensional node of the tree. If the height of P is i then x_i is inserted in P with a chain of EQSON-nodes storing $x_{i-1}, \ldots, x_1$. Any height-gap structure violations created as a result of the insertion are directly supported. The insertion of x_i into P may cause P to "overflow" in which case the tree needs to be rebalanced. If the height of P is greater than i then the empty subtree indicating the absence of x_i is replaced by a chain of EQSON-nodes storing $x_i, \ldots, x_1$. Any height-gap structure violation at the node storing x_i will be supported since the structure violation simply replaces a height-gap structure violation already existing in the tree, which must be supported.

Figure 2 illustrates SPLIT, the restructuring operation used in the rebalancing of a k-dimensional (a, b)-tree after the insertion of a k-dimensional vector. The rebalancing is initiated at an "overflown" node P (with b keys, indicated by an '*' beside the node) of height h. It involves the splitting of the node as is the case in an (a, b)-tree. In Figure 2.a P is split at its middle key, a. If P is not an EQSON node and has a parent, Q, of height $h + 1$ then a moves to Q and may possibly overflow Q; otherwise, a is stored in a new node, Q at height $h + 1$, and does not cause any new structure violation. If P is an EQSON node then SPLIT causes the creation of a new EQSON node, Q, at height $h + 1$. If the height of the parent of Q is greater than $h + 1$ then no further rebalancing needs to be performed. If, however, the height of Q is the same as that of its parent then another SPLIT needs to be performed.

Figure 2.b shows the application of SPLIT when the height of the EQSON subtree of key a in P is the same as the height of P. In this case P is split at a which is not necessarily the middle key of P. The operation causes the movement of a to the parent of P at height $h + 1$ or the creation of a new node, R, at height $h + 1$, storing a.

Theorem 4.1: A k-dimensional vector can be inserted in an (a, b)-tree of size n in $O(\log_a n + k)$ time. ■

The deletion of a k-dimensional vector proceeds with a search which ends in an i-dimensional node P such that x_i is found in P with a chain of EQSON-nodes storing $x_{i-1}, \ldots, x_1$ as the EQSON-subtree of x_i. If the left (or right) subtree of x_i is empty then x_i along with the empty subtree and the EQSON-subtree of x_i are deleted from P. If both the left and right subtrees of x_i are non-empty then the predecessor a_i of x_i is searched and found in an i-dimensional node Q. Next, x_i along with its EQSON-subtree are replaced by a_i along with its EQSON-subtree; a_i (in node Q) along with its right (empty) subtree and its EQSON-subtree are deleted from Q. In either case the deletion results in the decrease of the number of keys in a node, say P, by one. Rebalancing is needed only if the number of keys in P has become less than a - 1. If P is the root of the tree then either no rebalancing is needed or if the root has become unary (zero keys) then it is removed reducing the height of the tree by one. If the number of keys in P (not an EQSON node or the root) is reduced to < a - 2 (but not zero) then no rebalancing is needed since a shallow-node violation at P was already supported. Similar comments apply for an already existing height-gap structure violation.

If P is an EQSON node and the deletion resulted in a decrease in the height of the EQSON subtree rooted at P then any structure violations at the neighbors of P heretofore supported must be examined for loss of support.

The restructuring operations used in the cases described above are illustrated in Figure 3. The SHIFT operation (Figure 3.a) is performed if the total number of keys in P and Q (both at height h - 1) is at least 2a - 2. Keys (and corresponding subtrees) from Q are shifted to P (via R) such that P has a - 1 keys.

The MERGE operation (Figure 3.b) is applied if the total number of keys in P and Q is less than 2a - 2. After the MERGE, P may have less than a - 1 keys but only if Q had less than a - 1 keys before the MERGE (a shallow node violation at Q was already supported). DEMOTE (Figure 3.c) is actually a form of the MERGE operation in which there is a hypothetical node P at height h - 1 with zero keys. As a result of the MERGE or DEMOTE, the number of keys in R is reduced by one which may prompt further rebalancing.

Theorem 4.2: A k-dimensional vector can be deleted from an (a, b)-tree of size n in $O(\log_a n + k)$ time. ■

5. Amortized Analysis of Multidimensional (a, b)-trees, b ≥ 2a

Multidimensional (a, b)-trees, b ≥ 2a, are also called multidimensional weak (a, b)-trees. The model we use for the amortized analysis is the banker's view of amortization [18, 24]. As an accounting device, we assume that in a k-dimensional weak (a, b)-tree, non-negative credits are stored in nodes and EQSON-edges. For an extended treatment see [29].

Theorem 5.1: The rebalancing time for a mixed sequence of m insertions and deletions in a k-dimensional weak (a, b)-tree of initial size n is $O(k(m + n))$. ■

Corollary 5.1.1: The amortized rebalancing time for a mixed sequence of m insertions and deletions in a k-dimensional weak (a, b)-tree of initial size n is $O(1)$ if $m = \Omega(n)$ or if $n = 0$. ■

6. Self-Organizing (a, b)-trees

A (1-dimensional) *self-organizing (a, b)-tree* for a set of n (linearly ordered) keys, x_i with frequency of access w_i, $1 \leq i \leq n$, is a 2-dimensional (a, b)-tree such that the EQSON subtree of each 2-dimensional node N_i storing x_i is a "virtual" (a, b)-tree of size w_i with maximum possible height, i.e., $1 + \log_a((w_i + 1)/2)$.

Theorem 6.1: The access time of a self-organizing (a, b)-tree is logarithmic.

Proof: Let $A(i)$ be the access time of x_i. $A(i) = O(\log_a ((W + 1)/2) + 1/\log_2 a + 2 - (1 + \log_a ((w_i + 1)/2))) = O(\log_a W/w_i)$, where W is the sum of all access frequencies (see Theorem 3.5). ■

An access to x_i increases the frequency of x_i by 1. This can increase the height of the EQSON subtree of x_i by 1 and rebalancing is performed as in an insertion. Let the deletion of a key with frequency of access of 1 be called *special deletion*. It is treated similarly as deletion in a 2-dimensional (a, b)-tree.

Theorem 6.2: The amortized rebalancing time for a mixed sequence of m accesses, insertions, and special deletions in a self-organizing (a, b)-tree initially storing n items, $b \geq 2a$, is $O(n + m)$.

Proof: Follows from Theorem 5.1. ■

Corollary 6.2.1: The amortized rebalancing time for a mixed sequence of m accesses, insertions, and special deletions in a self-organizing (a, b)-tree, $b \geq 2a$, initially storing n items is $O(1)$ if $m = \Omega(n)$ or if $n = 0$. ■

7. Concluding Remarks

In this paper, we have presented the neighbor-support paradigm and used it to define multidimensional (a, b)-trees. We have shown that the resulting data structure has many desirable properties. It involves almost the same restructuring as in the base structure, i.e., (a, b)-trees, and has rather strong amortized rebalancing time-complexity results. We have also sketched the use of a new approach for height-analysis of neighbor-supported multidimensional trees to show an upper bound on the height of a k-dimensional (a, b)-tree that is within an additive factor of the corresponding upper bound on (a, b)-trees.

The k-dimensional (a, b)-trees that we have defined and investigated are important in their own right. They can form the basis of file structures for very large databases to support multi-key or associative queries (see [13]). The simplicity of the update algorithms and other desirable properties of these trees make them distinct from other attempts (for example, [8, 15]) at generalizing (a, b)-trees (or B-trees) to higher dimensions.

REFERENCES

[1] Adel'son-Vel'skii, G.M. and Landis, E.M., "An algorithm for the organization of information", *Doklady Akademii Nauk SSSR*, Tom 146, Nos. 1-6, 1962, pp. 263-266, tr. in *Soviet Mathematics*, Vol. 3, No. 5, September-October 1962, pp. 1259-1263.

[2] Bayer, R., "Symmetric binary B-trees: data structure and maintenance algorithms", *Acta Informatica*, Vol. 1, 1972, pp. 290-306.

[3] Bent, S.W., Sleator, D.D., and Tarjan, R.E., "Biased search trees", *SIAM Journal on Computing*, Vol. 14, pp. 545-568.

[4] Bentley, J.L. and Saxe, J.B., "Algorithms on vector sets", *SIGACT News*, Vol. 11, No. 2, Fall 1979, pp. 36-39.

[5] Chiang, Y.-J. and Tamassia, R., "Dynamic algorithms in computational geometry", *Department of Computer Science Technical Report No. CS-91-24*, Brown University, March 1991.

[6] Feigenbaum, J. and Tarjan, R.E., "Two new kinds of biased search trees", *Bell Systems Technical Journal*, Vol. 62, 1983, pp. 3139-3158.

[7] Gonzalez, T.F., "The on-line d-dimensional dictionary problem", (unpublished manuscript), Department of Computer Science, Utrecht University, October 1990.

[8] Gueting, R.H. and Kriegel, H.P., "Multidimensional B-tree: an efficient dynamic file structure for exact match queries", *Proceedings of the 10th GI Annual Conference Informatik Fachberichte*, Springer-Verlag, 1980, pp. 375-388.

[9] Gueting, R.H. and Kriegel, H.P., "Dynamic k-dimensional multiway search under time-varying access frequencies", *Proceedings of the 5th GI Conference on Theoretical Computer Science*, Springer-Verlag, 1981, pp. 135-145.

[10] Guibas, L.J. and Sedgewick, R., "A dichromatic framework for balanced trees", *19th Annual Symposium on Foundations of Computer Science*, Long Beach, California: IEEE Computer Society, 1978, pp. 8-21.

[11] Hirschberg, D.S., "On the complexity of searching a set of vectors", *SIAM Journal on Computing*, Vol. 9, 1980, pp. 126-129.

[12] Iyengar, S.S., Rao, N.S.V., Kashyap, R.L. and Vaishnavi, V.K., "Multidimensional data structures: review and outlook", in *Advances in Computers* (ed. M. Yovits), Vol. 27, 1988, pp. 69-119.

[13] Knuth, D.E., *The Art of Computer Programming*, Vol 3: Sorting and Searching, Addison-Wesley, Reading, MA, 1973.

[14] Kriegel, H.P., "Performance comparison of index structures for multi-key retrieval", *Proceedings of the Annual Meeting of SIGMOD*, 1984, pp. 186-195.

[15] Kriegel, H.P. and Vaishnavi, V.K., "Multidimensional leaf B-trees", *Computer Science Technical Report No. 109*, Universität Dortmund, West Germany, 1980.

[16] Kriegel, H.P. and Vaishnavi, V.K., "A nearly optimal dynamic tree structure for partial-match queries with time-varying frequencies", *Proceedings of the Annual Conference on Information Sciences and Systems*, 1981, pp. 193-197.

[17] Mehlhorn, K., "Dynamic binary search", *SIAM Journal on Computing*, Vol. 8, 1979, pp. 175-198.

[18] Mehlhorn, K., *Data Structures and Algorithms*, Vol 1: Sorting and Searching, Springer-Verlag, 1984.

[19] Mehlhorn, K., *Data Structures and Algorithms*, Vol 2: Multidimensional Tree Structures and Computational Geometry, Springer-Verlag, 1984.

[20] Overmars M.H., *The Design of Dynamic Data Structures*, No. 156 in Lecture Notes in Computer Science (ed. Goos, G. and Hartmanis, J.), Berlin: Springer-Verlag, 1983.

[21] Scheuermann, P. and Ouksel, M., "Multidimensional B-trees for associative searching in database systems", *Information Systems*, Vol. 7, 1982, pp. 123-137.

[22] Sleator, D.D. and Tarjan, R.E., "Self-adjusting binary search trees", *Journal of the ACM*, Vol. 32, 1985, pp. 652-686.

[23] Tarjan, R.E., "Updating a balanced search tree in O(1) rotations", *Information Processing Letters*, Vol. 16, 1983, pp. 253-257.

[24] Tarjan, R.E., "Amortized computational complexity", *SIAM Journal on Algebraic and Discrete Methods*, Vol. 6, No. 2, April 1985, pp. 306-318.

[25] Vaishnavi, V.K., "Multidimensional height-balanced trees", *IEEE Transactions on Computers*, Vol. C-33, No. 4, April 1984, pp. 334-343.

[26] Vaishnavi, V.K., "Weighted leaf AVL-trees", *SIAM Journal on Computing*, Vol. 16, No. 3, June 1987, pp. 503-537.

[27] Vaishnavi, V.K., "ERRATUM: Weighted leaf AVL-trees", *SIAM Journal on Computing*, Vol. 19, No. 3, June 1990, p. 591.

[28] Vaishnavi, V.K., "Multidimensional balanced binary trees", *IEEE Transactions on Computers*, Vol. 38, No. 7, July 1989, pp. 968-985.

[29] Vaishnavi, V.K., "On *k*-dimensional balanced binary trees", *Computer Information Systems Technical Report No. CIS-90-001*, Department of Computer Information Systems, Georgia State University, January 1990.

[30] Vaishnavi, V.K. and Korson, T.D., "Height-analysis of *k*-dimensional leaf and node height-balanced trees: a new approach", *BIT*, Vol. 31, 1991, pp. 36-49.

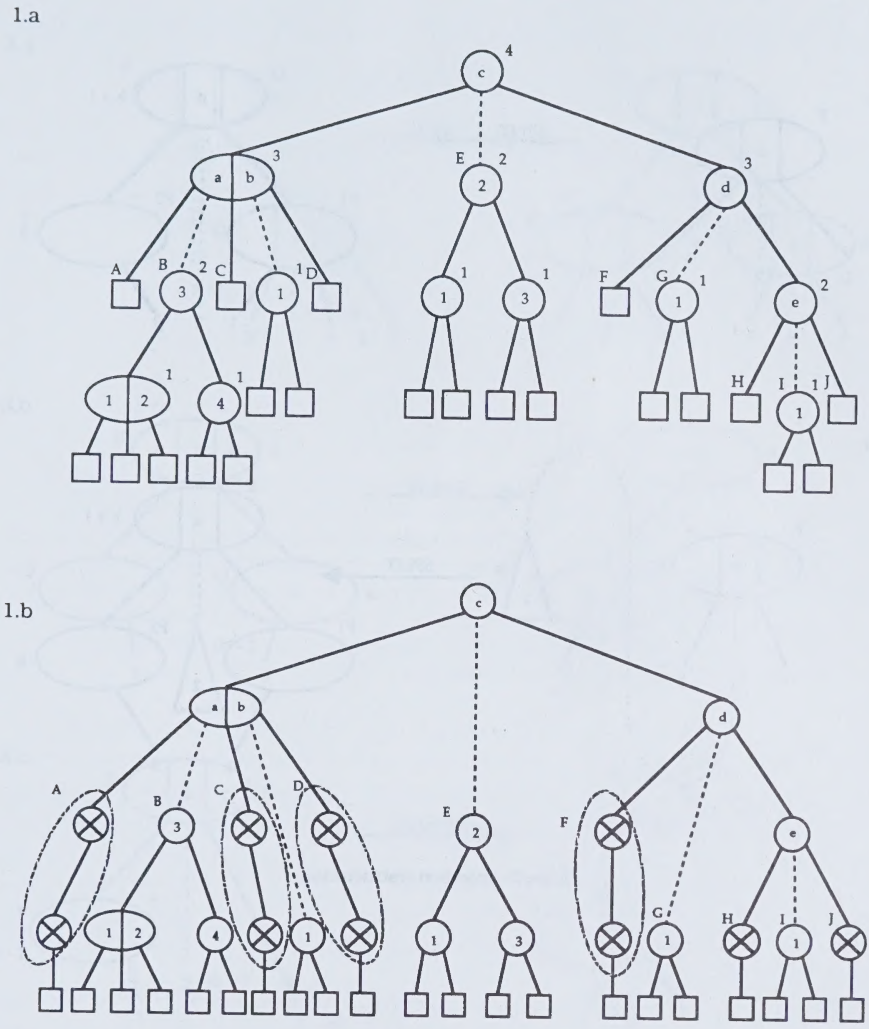

1.a
c
4
a
b
3
E
2
2
d
3
A
B
2
C
1
D
3
1
1
1
1
3
F
G
1
1
e
2
1
2
1
4
1
H
I
1
J
1
1.b
c
a
b
d
A
B
C
D
3
E
2
F
e
1
2
4
1
1
3
G
1
H
I
1
J

Figure 1. A 2-dimensional ternary node search tree which is also a 2-dimensional (2,3)-tree containing the following set of 2-dimensional vectors:

{(a,1), (a,2), (a,3), (a,4), (b,1), (c,1), (c,2), (c,3), (d,1), (e,1)}.

In 1.a the height of each internal node is shown beside the node. 1.b is another form of 1.a in which missing nodes at various heights, other than EQSON nodes, are shown as unary nodes.

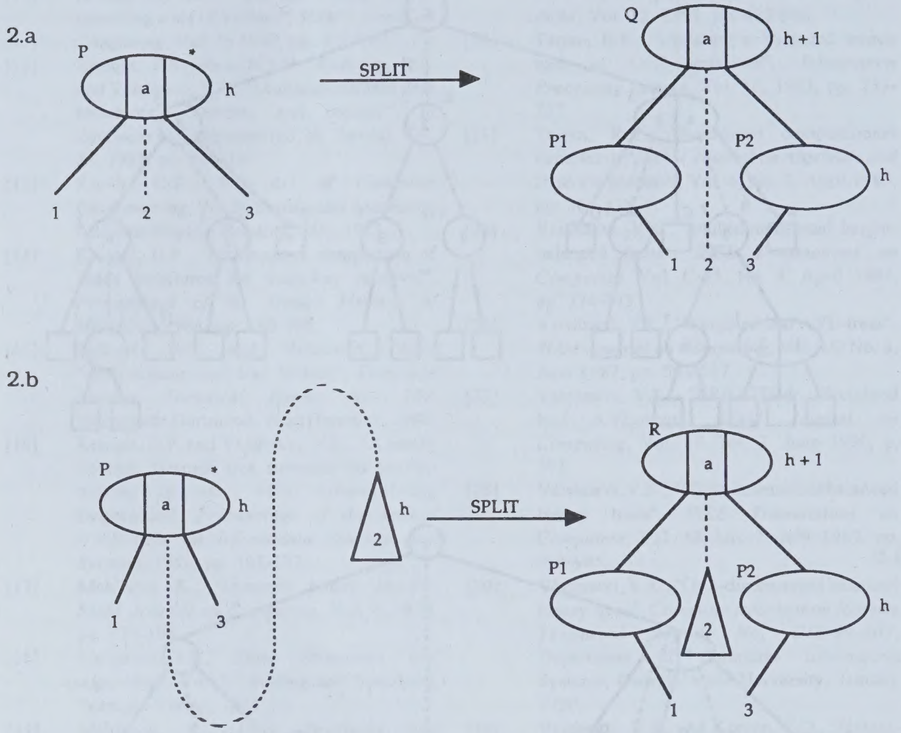

2.a
P
*
a
h
1
2
3
SPLIT
Q
a
h + 1
P1
P2
h
1
2
3
2.b
P
*
a
h
1
3
h
2
SPLIT
R
a
h + 1
P1
P2
h
2
1
3

Figure 2. Insertion restructuring.

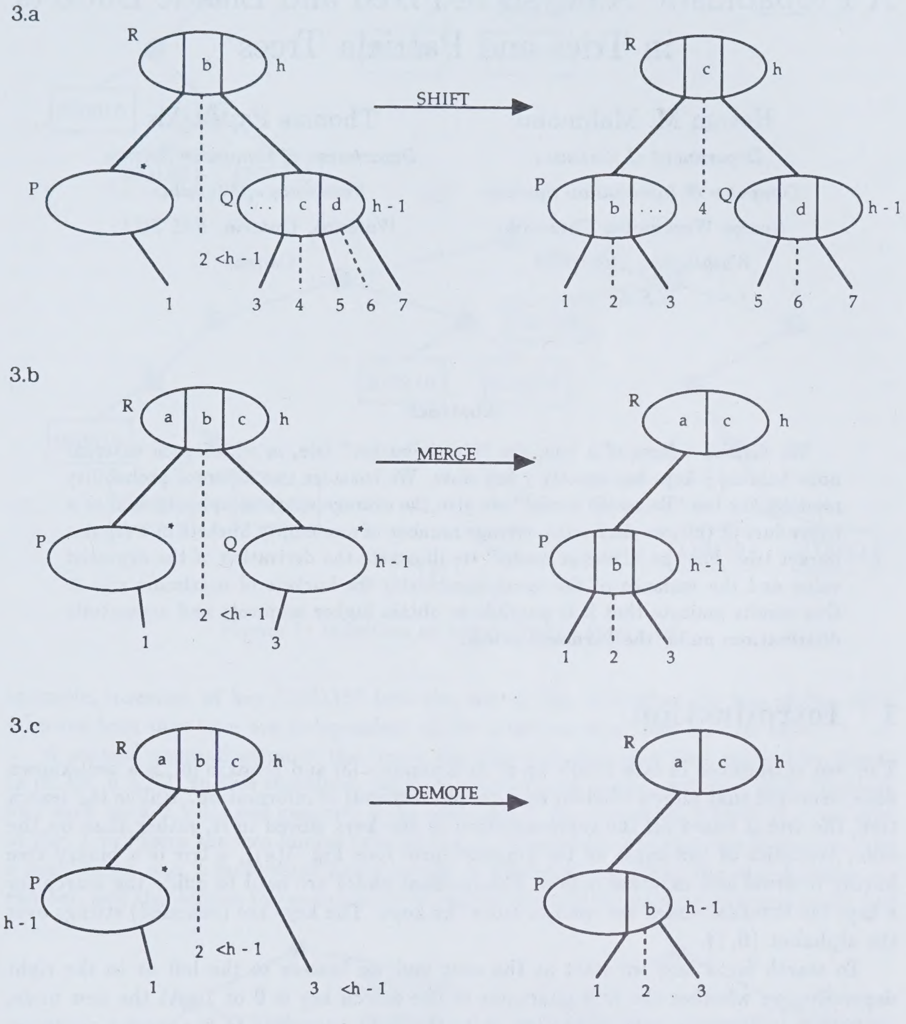

3.a
R
b
h
SHIFT
P
*
Q
...
c
d
h - 1
2 <h - 1
1
3
4
5
6
7
R
c
h
P
b
...
4
Q
d
h - 1
1
2
3
5
6
7
3.b
R
a
b
c
h
MERGE
P
*
Q
*
h - 1
2 <h - 1
1
3
R
a
c
h
P
b
h - 1
1
2
3
3.c
R
a
b
c
h
DEMOTE
P
*
h - 1
2 <h - 1
1
3 <h - 1
R
a
c
h
P
b
h - 1
1
2
3

Figure 3. Deletion Restructuring

A Probabilistic Analysis of Fixed and Elastic Buckets in Tries and Patricia Trees*

Hosam M. Mahmoud
Department of Statistics/
Computer & Information Systems
George Washington University
Washington, DC 20052
U.S.A.

Thomas Papadakis
Department of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1
Canada

Abstract

We discuss a form of a trie, the "elastic-bucket" trie, in which each external node holding j keys has exactly j key slots. We consider two different probability models. For the "Bernoulli model" we give the average space complexity, and as a byproduct of this, we derive the average number of non-empty buckets in a regular bucket trie. For the "Poisson model" we illustrate the derivation of the expected value and the variance of the space complexity for buckets of maximum size 3. Our results indicate that it is possible to obtain higher moments and asymptotic distributions under the Bernoulli model.

1 Introduction

The *trie*, introduced in late 1950's by de la Briandais [3] and Fredkin [6], is a well-known data structure that allows efficient update and re*trie*val of information. Unlike the search tree, the trie is based on the representation of the keys stored in it, rather than on the order statistics of the keys. In its simplest form (see Fig. 1(a)), a trie is a binary tree having internal and external nodes. The internal nodes are used to guide the search for a key; the external nodes are used to store the keys. The keys are (encoded) strings over the alphabet $\{0, 1\}$.

To search for a key, we start at the root and we branch to the left or to the right depending on whether the first character of the search key is 0 or 1. At the new node, we branch in the same way to the left or to the right according to the second character of the key, etc. When we reach an external node, we compare the search key with the key (if any) stored in that external node.

To insert a new key, we first search for it. If our unsuccessful search terminates in an empty external node, we simply insert the new key in that node. If our search terminates in a non-empty external node, we extend the search path by adding an appropriate number of internal nodes, and we finally store the new key in a new external node. For

---

*This research was supported in part by the Natural Science and Engineering Research Council of Canada under grant No. A-8237, and the Information Technology Research Centre of Ontario.

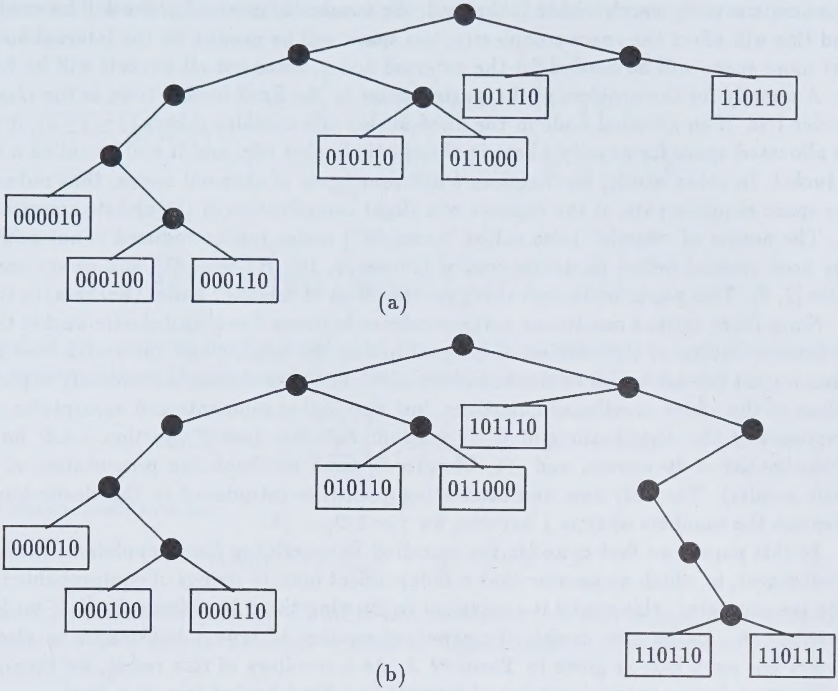

000010
000100
000110
010110
011000
101110
110110
(a)
000010
000100
000110
010110
011000
101110
110110
110111
(b)

Figure 1: Insertion of key "110111" into a trie

example, insertion of key "110111" into the trie of Fig. 1(a) gives the trie of Fig. 1(b). Observe here that tries are independent of the insertion sequences of their keys.

A natural generalization of the trie is the *(fixed-)bucket trie*, first studied by Knuth in [8, Ex. 6.3.20]. In this structure, we store up to b keys in each external node (*bucket*), and each bucket has fixed capacity b, for some $b > 1$. For example, if $b = 2$, the two tries of Fig. 1 will form the two bucket tries of Fig. 2. Obviously, a bucket trie is shorter than a non-bucket one, a fact which has a couple of implications. First, the search paths are shorter, and this affects the search time; it will be cheaper to reach an external node, but

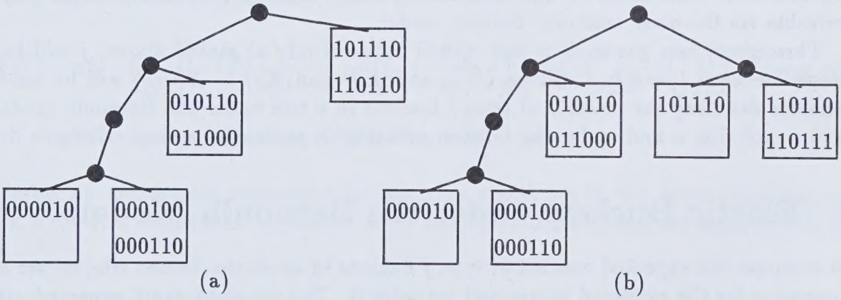

000010
000100
000110
010110
011000
101110
110110
(a)
000010
000100
000110
010110
011000
101110
110110
110111
(b)

Figure 2: Insertion of key "110111" into a fixed-bucket trie

more expensive to search within it. Second, the number of internal nodes will be smaller, and this will affect the space complexity; less space will be needed for the internal nodes, but more space will be needed for the external nodes, since not all buckets will be full.

A remedy for the problem of the wasted space in the fixed-bucket tries, is the *elastic-bucket trie.* If an external node in the fixed-bucket trie contains j keys $(1 \leq j \leq b)$, it will be allocated space for exactly j keys in the elastic-bucket trie, and it will be called a *type j bucket.* In other words, we maintain b different types of external nodes, thus reducing the space requirements, at the expense of a slight complication in the update algorithms.

The notion of "elastic" (also called "exact-fit") nodes just introduced is not new. It has been studied before in the context of B-trees [5, 10], B$^+$-trees [1], and m-ary search trees [2, 9]. This paper is the first study on the effect of "elastic" nodes (buckets) in tries.

Since there exists a one-to-one correspondence between fixed- and elastic-bucket tries, all known results on the number of internal nodes, the height, and the search cost in a fixed-bucket trie carry over to elastic-bucket tries. These results include not only expected values of the above mentioned quantities, but also higher moments and asymptotic convergences of the distributions of these random variables (see [7, Section 3.4.4] for an extensive list of 46 papers, and [11, Chapter 5] for a textbook-like presentation of the basic results). The only new and interesting quantities introduced in the elastic-bucket tries are the numbers of type j buckets, for $j = 1, 2, \ldots, b$.

In this paper, we first consider the so called *Bernoulli* (or *fixed population*) *model* of randomness, in which we assume that n independent infinite strings of equiprobable i.i.d. bits are generated; this model is equivalent to drawing the n keys from the [0,1]-uniform distribution. Under this model, the expected number of type j buckets in an elastic-bucket trie on n keys is given in Theorem 2. As a corollary of this result, we then give in Theorem 3 the expected number of buckets in a fixed-bucket trie on n keys.

Difficulties related to the derivation of higher moments of the number of buckets, caused us to consider next the *Poisson model* of randomness. In this model, we assume that the number of keys in the trie is not fixed any more, but it is a random variable following the Poisson distribution with mean z. The Poisson model in the context of tries was introduced in 1979 by Fagin, Nievergelt, Pippenger and Strong [4], and in addition to being an interesting model, it can also be viewed as a mathematical transformation that facilitates the derivation of certain results, which can be transformed back into results under the (original) Bernoulli model. In this paper, we illustrate the derivation of the expected values and the covariance matrix between the numbers of different bucket types in a trie, by considering the case $b = 3$. The variances and covariances are all $O(z)$, and the expected values are the same as those under the Bernoulli model with fixed population n. This suggests that additional results under the Bernoulli model may be derivable via the more tractable Poisson model.

Throughout our paper, b, n and z, will be used only as stated above, j will be an integer between 1 and b, i will be $\sqrt{-1}$, and $X_n^{(j)}$ and $X^{(j)} = X^{(j)}(z)$ will be random variables denoting the number of type j buckets in a trie under the Bernoulli model of fixed population n and under the Poisson model with parameter z respectively.

2 Elastic Buckets under the Bernoulli Model

To compute the expected number of type j buckets in an elastic-bucket trie, we set up a recurrence for the expected values and we solve it. The obtained exact expected values involve an alternating sum, which we evaluate by applying Rice's method.

Theorem 1 *Under the Bernoulli model, the expected number of type j buckets in a trie on n keys is*

$$\mathbf{E}(X_n^{(j)}) = \begin{cases} n - \sum_{k=2}^{n} (-1)^k \binom{n}{k} g_1(k) & \text{if } j = 1 \\ (-1)^j \sum_{k=2}^{n} (-1)^k \binom{n}{k} g_j(k) & \text{if } 2 \le j \le b \end{cases}$$

where

$$g_j(k) \overset{\text{def}}{=} \frac{\binom{k}{j}}{1 - 2^{1-k}} \left(1 - \frac{1}{2^{j-1}} \sum_{r=0}^{b-j} (-1)^r \binom{k-j}{r} \frac{1}{2^r} \right) . \tag{1}$$

Proof: If we condition on the number of keys in the left subtree, and we observe that this number follows the binomial distribution with parameter 1/2, we get

$$\mathbf{E}(X_n^{(j)}) = \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} \mathbf{E}(X_k^{(j)}), \qquad \text{for} \quad n > b.$$

The initial conditions are

$$\mathbf{E}(X_n^{(j)}) = \delta_{n,j}, \qquad \text{for} \quad 0 \le n \le b,$$

where $\delta_{n,j}$ is Kronecker's delta. To solve this recurrence, we may appeal directly to the solution of a more general recurrence, given by Szpankowski in [12, equations (3.5), (3.7) and (3.12)] to get

$$\mathbf{E}(X_n^{(j)}) = n\delta_{1,j} + \sum_{k=2}^{n} (-1)^k \binom{n}{k} \frac{-\sum_{r=0}^{b} (-1)^r \binom{k}{r} r! \beta_{r,j}}{1 - 2^{1-k}} , \quad \text{for} \quad n \ge 1. \tag{2}$$

as its solution, where the $\beta_{r,j}$'s can be simplified to

$$\beta_{0,j} = 0 \tag{3}$$

and

$$\beta_{r,j} = -\frac{1}{r!} \left(\delta_{r,j} - \frac{1}{2^{r-1}} \binom{r}{j} \right) , \quad \text{for} \quad 1 \le r \le b. \tag{4}$$

Putting together now (2), (3) and (4), we get

$$\mathbf{E}(X_n^{(j)}) = n\delta_{1,j} + \sum_{k=2}^{n} \left\{ (-1)^k \binom{n}{k} \frac{1}{1 - 2^{1-k}} \left((-1)^j \binom{k}{j} - \sum_{r=1}^{b} (-1)^r \binom{k}{r} \frac{1}{2^{r-1}} \binom{r}{j} \right) \right\}, \quad \text{for } n \ge 1,$$

which simplifies to the expression sought. ■

To derive an asymptotic expression for the sum appearing in the above theorem, we follow a method introduced by Knuth in [8, Exercise 5.2.2.53], and attributed to S. O. Rice.

Theorem 2 *Under the Bernoulli model, the expected number of type j buckets in a trie on n keys is*

$$\mathbf{E}(X_n^{(j)}) = \begin{cases} n\left(1 - \dfrac{1}{\ln 2}\displaystyle\sum_{r=1}^{b-1}\frac{1}{r2^r}\right) + n\delta(\lg n) + O(1) & \text{if } j=1 \\ \dfrac{n}{j(j-1)\ln 2}\left(1 - \dfrac{1}{2^{j-1}}\displaystyle\sum_{r=0}^{b-j}\binom{j-2+r}{r}\frac{1}{2^r}\right) + n\delta(\lg n) + O(1) & \text{if } 2 \le j \le b \end{cases} \quad \text{as } n \to \infty, \tag{5}$$

where $\delta(\lg n)$ is a periodic function of $\lg n$ with small magnitude.

Proof: According to Theorem 1, Rice's formula yields

$$\mathbf{E}(X_n^{(j)}) = n\delta_{1,j} - \frac{1}{2\pi i}\oint_C B(n+1,-s)\, g_j(s)\, ds, \tag{6}$$

where $B(n+1,-s)$ is the classical Beta function, $g_j(s)$ is defined as in (1), and C is a rectangle encompassing the integer points $2, 3, \ldots, n$. Consider now a larger rectangle C', with corners at the four points $(-\frac{1}{2} \pm iN,\ M \pm iN)$, for $M > n$. By using the classical approximations

$$\frac{\Gamma(s+a)}{\Gamma(s+b)} = s^{a-b} + O(s^{a-b-1}) \qquad \text{as} \quad s \to \infty \tag{7}$$

and

$$\Gamma(x+iy) = O(x^{x-\frac{1}{2}}e^{-\frac{\pi y}{2}}) \qquad \text{as} \quad y \to \infty,$$

we can see that the integral of $B(n+1,-s)\, g_j(s)$ vanishes along the top and bottom lines of C' as $N \to \infty$, and along the right line of C' as $M \to \infty$. On the left line of C', this integral is $O(1)$. From Cauchy's Theorem now on complex functions, it follows that $-\frac{1}{2\pi i}\oint_C B(n+1,-s)\, g_j(s)\, ds$ equals the sum of residues of the integrand at the poles that lie outside C but inside C' when $N, M \to \infty$. In this region, $B(n+1,-s)$ has poles at $s=0$ and $s=1$, and $g_j(s)$ has poles at $s = 1 \pm i\frac{2k\pi}{\ln 2}$, for $k = 0, 1, 2, \ldots$. Since $g_j(s)$ can always be written as $s(s-1)h(s)$, the integrand $B(n+1,-s)\, g_j(s)$ has simple poles at $s_0 = 1$ and $s_k = 1 + i\frac{2k\pi}{\ln 2}$, for $k = \pm 1, \pm 2, \ldots$. The poles at s_k contribute only $n\delta(\lg n)$, as can be easily verified. The only important contribution comes from the pole at $s_0 = 1$. So, we have

$$-\frac{1}{2\pi i}\oint_C B(n+1,-s)\, g_j(s)\, ds = \operatorname*{Res}_{s=1}\{B(n+1,-s)\, g_j(s)\} + n\,\delta(\lg n) + O(1). \tag{8}$$

The residue in the last equality can be written as

$$\frac{\Gamma(n+1)}{\Gamma(n)}\,\frac{1}{\ln 2}\,\lim_{s\to 1}\left\{\frac{\dbinom{s}{j}\left(1 - \dfrac{1}{2^{j-1}}\displaystyle\sum_{r=0}^{b-j}(-1)^r\binom{s-j}{r}\frac{1}{2^r}\right)}{s-1}\right\},$$

which, according to (7), and after doing some routine calculations, becomes

$$\begin{cases} (n+O(1))\dfrac{1}{\ln 2}\displaystyle\sum_{r=1}^{b-1}\frac{1}{r2^r} & \text{for} \quad j=1 \\ (n+O(1))\dfrac{1}{\ln 2}(-1)^j\dfrac{1}{j(j-1)}\left(1 - \dfrac{1}{2^{j-1}}\displaystyle\sum_{r=0}^{b-j}\binom{j-2+r}{r}\frac{1}{2^r}\right) & \text{for} \quad 2 \le j \le b \end{cases}.$$

The theorem now follows from (6) and (8). ■

3 Fixed Buckets under the Bernoulli Model

To derive $\mathbf{E}(X_n)$, the expected number of buckets in a fixed-bucket trie, we can work as in the previous section. We can set up the recurrence

$$\mathbf{E}(X_n) = \frac{1}{2^{n-1}} \sum_{k=0}^{n} \binom{n}{k} \mathbf{E}(X_k) \qquad \text{for} \quad n > b$$

with initial conditions $\mathbf{E}(X_0) = 0$ and $\mathbf{E}(X_1) = \mathbf{E}(X_2) = \cdots = \mathbf{E}(X_b) = 1$, solve it to obtain an exact expression for $\mathbf{E}(X_n)$, and apply Rice's method again to evaluate the resulting alternating sum. However, having derived $\mathbf{E}(X_n^{(j)})$ in the previous section, we can derive $\mathbf{E}(X_n)$ in a less tedious way, since $\mathbf{E}(X_n) = \sum_{j=1}^{b} \mathbf{E}(X_n^{(j)})$.

Theorem 3 *Under the Bernoulli model, the expected number of (non-empty) fixed buckets in a trie on n keys is*

$$\mathbf{E}(X_n) = n - \sum_{k=2}^{n} (-1)^k \binom{n}{k} \frac{\sum_{r=2}^{b} \left\{ (-1)^r \binom{k}{r} \left(1 - \frac{1}{2^{r-1}}\right) \right\}}{1 - 2^{1-k}} \tag{9}$$

or

$$\mathbf{E}(X_n) = n \left\{ 1 - \frac{1}{\ln 2} \left(1 - \frac{1}{b} - \sum_{r=2}^{b} \frac{1}{r(r-1)2^{r-1}} \right) \right\} + n\,\delta(\lg n) + O(1) \quad \text{as} \quad n \to \infty, \tag{10}$$

where $\delta(\lg n)$ is a periodic function of $\lg n$ with small magnitude.

Proof: Omitted. ■

It is worth mentioning here that the above result does not follow from the well-known

$$\frac{n}{b \ln 2} + n \text{ oscillations } + O(1)$$

(see [8, Exercise 6.3.20]) expected number of internal nodes under the Bernoulli model; recall that we are not counting the "empty" buckets.

Before we proceed, we may also note here that although no closed formula exists for the sum appearing in

$$c_b \overset{\text{def}}{=} 1 - \frac{1}{\ln 2} \left(1 - \frac{1}{b} - \sum_{r=2}^{b} \frac{1}{r(r-1)2^{r-1}} \right),$$

the coefficient of n in the expected number of fixed buckets, this sum converges very fast; for example, for all $b \geq 8$, it is .3068, computed to 4-digit accuracy.

An interesting question that we are now in a position to answer is the following: how much bucket space is unused in a fixed-bucket trie? We know that the used space (number of filled key slots) is exactly n. Therefore, it suffices to compute the total space nbc_b. The next corollary shows that the bucket space utilization is about 69%.

Corollary 1 *Under the Bernoulli model, the expected space taken by all (non-empty) buckets in a fixed-bucket trie on n keys is proportional to n as $n \to \infty$, with a proportionality factor*

$$bc_b = \frac{1}{\ln 2} + O\left(\frac{1}{b2^b}\right) \approx 1.44269 + O\left(\frac{1}{b2^b}\right) \qquad \text{as} \quad b \to \infty.$$

Proof: By rewriting the finite sum appearing in c_b as the difference of two infinite sums, we get

$$bc_b = \frac{1}{\ln 2} + b\left(1 - \frac{1}{\ln 2} + \frac{1}{\ln 2}\left(\sum_{r\geq 2}\frac{1}{r(r-1)2^{r-1}} - \sum_{r\geq b+1}\frac{1}{r(r-1)2^{r-1}}\right)\right).$$

The result now follows by observing that the first sum in the last expression is $1 - \ln 2$, and the second sum is $O(b^{-2}2^{-b})$ as $b \to \infty$. ■

4 The Poisson Model

In this section, we illustrate the derivation of the expected values and the variances of the numbers of buckets in a trie, under the Poisson model, by considering the special case $b = 3$. Our results can be generalized for arbitrary b.

Consider the random variables

$$X = X(z) = \alpha_1 X^{(1)} + \alpha_2 X^{(2)} + \alpha_3 X^{(3)}$$

and

$$X_n = \alpha_1 X_n^{(1)} + \alpha_2 X_n^{(2)} + \alpha_3 X_n^{(3)},$$

where $\alpha_1, \alpha_2, \alpha_3$ are arbitrary non-negative real numbers. Clearly, if we can compute $\mathbf{E}(X)$, we can then compute the expected number of buckets of any type, and if compute $\mathbf{Var}(X)$, we can subsequently compute the variance-covariance matrix for the elastic buckets of all types.

We start by considering the moment generating functions $\Psi(z,u) = \mathbf{E}(e^{uX}) = \sum_{k\geq 0}\frac{\mathbf{E}(X^k)}{k!}u^k$ and $\Psi_n(u) = \mathbf{E}(e^{uX_n}) = \sum_{k\geq 0}\frac{\mathbf{E}(X_n^k)}{k!}u^k$ of X and X_n respectively, for which it is easy to verify

$$\Psi(z,u) = \sum_{n\geq 0}\Psi_n(u)\frac{z^n e^{-z}}{n!}. \tag{11}$$

We will use these generating functions to derive the expected value

$$\mathbf{E}(X) \overset{\text{def}}{=} A(z) = \frac{\partial \Psi(z,0)}{\partial u} \tag{12}$$

and the variance

$$\mathbf{Var}(X) \overset{\text{def}}{=} V(z) = \frac{\partial^2 \Psi(z,0)}{\partial u^2} - A^2(z). \tag{13}$$

First, we can derive $\Psi_n(u) = \frac{1}{2^n}\sum_{j=0}^{n}\left\{\mathbf{E}(e^{u(X_j+X_{n-j})})\binom{n}{j}\right\}$, for $n > 3$, by conditioning on the number of keys in the left subtree. We now multiply both sides of the last equality by $e^{-z}z^n$, and we add up all resulting equalities for all $n > 3$. If we use (11) and the initial conditions $\Psi_j(u) = e^{\alpha_j u}$, for $j = 1, 2, 3$, we arrive at the functional equation

$$\Psi(z,u) = \Psi^2\left(\frac{z}{2}, u\right) + e^{-z}z^2\left(\frac{e^{\alpha_2 u} - e^{2\alpha_1 u}}{4} + \frac{e^{\alpha_3 u} - e^{(\alpha_1+\alpha_2)u}}{8}\, z\right). \tag{14}$$

We can now use (14) and (12) to compute $\mathbf{E}(X)$, and subsequently (14), (12) and (13) to compute $\mathbf{Var}(X)$.

Theorem 4 *Under the Poisson model with parameter z, the expected value of the linear combination of the numbers of buckets of all types in a trie is*

$$\mathbf{E}(X(z)) = \frac{z}{\ln 2}\left(\alpha_1 \ln 2 + \frac{-5\alpha_1 + \alpha_2 + \alpha_3}{8}\right) + z\,\delta(\lg z) + O(1),$$

where $\delta(\lg z)$ is a periodic function of $\lg z$ with small magnitude.

Proof: By differentiating (14) once at $u = 0$, we get

$$A(z) = 2A\left(\frac{z}{2}\right) + e^{-z}z^2 h(z), \tag{15}$$

where

$$h(z) \overset{\text{def}}{=} \frac{\alpha_2 - 2\alpha_1}{4} + \frac{\alpha_3 - \alpha_2 - \alpha_1}{8}\, z. \tag{16}$$

An iterative solution of the last recurrence yields

$$A(z) = \lim_{k\to\infty}\left\{2^k A\left(\frac{z}{2^k}\right)\right\} + z^2 \sum_{k\geq 0}\left\{\left(\frac{1}{2}\right)^k e^{-z/2^k} h\left(\frac{z}{2^k}\right)\right\}. \tag{17}$$

Since $A(z) = \alpha_1 z e^{-z} + O(z^2)$ as $z \to 0$, it follows that the limit in (17) is $\alpha_1 z$. The sum in (17) is a *harmonic sum*, and it can be computed by applying the standard Mellin transform technique (see, for example, [11, Section 1.5]). This sum evaluates to

$$\frac{1}{8z\ln 2}(-5\alpha_1 + \alpha_2 + \alpha_3) + \frac{1}{z}\,\delta(\lg z) + O\left(\frac{1}{z^2}\right) \qquad \text{as} \quad z \to \infty,$$

which completes the proof. ■

Theorem 5 *Under the Poisson model with parameter z, the variance of the linear combination of the numbers of buckets of all types in a trie is*

$$\begin{aligned}\mathbf{Var}(X(z)) \;&=\; \frac{z}{\ln 2}\left\{\alpha_1^2 \ln 2 + \frac{127\alpha_1^2 + 33\alpha_2^2 + 35\alpha_3^2 - 68\alpha_1\alpha_2 - 146\alpha_1\alpha_3}{256}\right.\\ &\qquad \left. + \frac{(\alpha_2 - 2\alpha_1)(3\alpha_3 - 7\alpha_1 - \alpha_2)}{16}\sum_{k\geq 0}\frac{4^k}{(2^k+1)^3} + \frac{3(\alpha_3 - \alpha_2 - \alpha_1)^2}{8}\sum_{k\geq 0}\frac{8^k}{(2^k+1)^5}\right\}\\ &\quad + z\,\delta(\lg z) + O(1) \qquad \textit{as} \quad z \to \infty,\end{aligned}$$

where $\delta(\lg z)$ is a periodic function of $\lg z$ with small magnitude.

Proof: By differentiating the functional equation (14) twice at $u = 0$, and by using (13) and (15), we get

$$V(z) = 2V\left(\frac{z}{2}\right) + C(z), \tag{18}$$

where

$$\begin{aligned}C(z) \;&\overset{\text{def}}{=}\; e^{-z}z^2 h(z) + 2e^{-z}z^2 h(z)A(z) + e^{-2z}z^4 h^2(z)\\ &\qquad + e^{-z}z^2\left(\frac{\alpha_2(\alpha_2 - 1) - 2\alpha_1(2\alpha_1 - 1)}{4} + \frac{\alpha_3(\alpha_3 - 1) - (\alpha_1 + \alpha_2)(\alpha_1 + \alpha_2 - 1)}{8}\, z\right),\end{aligned}$$

where $h(z)$ defined in (16). An iterative solution of the recurrence (18) yields

$$V(z) = \lim_{k\to\infty} \left\{ 2^k V\left(\frac{z}{2^k}\right) \right\} + \sum_{k\geq 0} 2^k C\left(\frac{z}{2^k}\right).$$

The limit in the above expression can be shown to be $\alpha_1^2 z$, and the harmonic sum there can be handled by the Mellin transform method. The only problem that arises is that $C(z)$ contains $A(z)$. Since the Mellin transform of $A(z)$ cannot be computed from its closed form given in Theorem 4, it is computed instead term-by-term from its representation in (17). The result follows after some lengthy but straightforward calculations. ■

We can now compute the complete variance-covariance matrix:

$$z \begin{pmatrix} 0.0956 & -0.0265 & 0.0185 \\ -0.0265 & 0.1888 & -0.0570 \\ 0.0185 & -0.0570 & 0.0919 \end{pmatrix} + Q(z),$$

where $Q(z)$ is a 3x3 residual matrix, whose (j,k)th entry is $z\delta_{jk}(z) + O(1)$, for $1 \leq j, k \leq 3$, and each $\delta_{jk}(z)$ is a periodic function of $\lg z$ with small magnitude. Furthermore, the variance of the number of fixed buckets can be seen to be approximately $0.2463z$ + oscillations $z + O(1)$. In view of the relatively small standard deviation, we may conclude that $X^{(j)}(z)$ is asymptotic *in probability* to $\mathbf{E}(X^{(j)})$ as $z \to \infty$, a consequence of Chebychev's inequality. Thus with very high probability $X^{(j)}$ stays close to its mean value if z is large.

5 Discussion

We analyzed the space complexity in the elastic- and the fixed-bucket trie. For the Bernoulli model of randomness (fixed number of keys, drawn from the [0,1]-uniform distribution), we derived the exact expected value, and we used Rice's method to evaluate the sums appearing in it. For the Poisson model of randomness (variable number of keys, following the Poisson distribution), we illustrated the derivation of the expected value and the variance by considering the case of a maximum bucket size of 3; we set up a functional equation for the moment generating function, we differentiated it, and we derived our results by means of Mellin transforms.

Our results hold for the fixed-bucket and elastic-bucket Patricia trees as well. To see why this is true, it suffices to recall that a Patricia tree is the same as a trie, but with all nodes of outdegree 1 eliminated. This implies that the pattern of the external nodes in both schemes is identical.

If we view the Poisson model of randomness as merely a mathematical tool for the derivation of results under the (more realistic) Bernoulli model, then our results under the Poisson model make us very optimistic about the derivation of higher moments and asymptotic distributions under the Bernoulli model. Indeed, we have shown (although not presented here) that, for general b, the random variable $Y = Y(z) = \alpha_1 X^{(1)} + \cdots + \alpha_b X^{(b)}$ has expected value

$$\begin{aligned} \mathbf{E}(Y(z)) &= z\left(\alpha_1 + \frac{1}{\ln 2}\sum_{k=2}^{b} c_k (k-2)!\right) + z\frac{2}{\ln 2}\sum_{k=2}^{b} c_k \sum_{j=1}^{\infty} \Re\left(\Gamma\left(k-1-i\frac{2j\pi}{\ln 2}\right) e^{i\,2j\pi\log_2 z}\right) \\ &\quad + O(z^{-d}), \qquad \text{for any} \quad d > -1, \end{aligned}$$

where

$$c_k = \frac{\alpha_k}{k!} - \frac{1}{k!2^{k-1}} \sum_{j=0}^{b} \binom{n}{j} \alpha_j, \qquad \text{for} \quad 2 \le k \le b.$$

A quick calculation now reveals that the expected values under the Poisson and the Bernoulli model are the same.

As a consequence of our results presented here, we observed that on the average, and under the Bernoulli model, only 69% of the bucket space is occupied in the fixed-bucket tries. Since, by definition, 100% of the bucket space is occupied in the elastic-bucket tries, it was natural to ask the following question. If only two bucket sizes are allowed, b and say x, then what should x be, so that the wasted bucket space will be minimized? Our experimental results show that the best choice for x is about $.7b$, in which case about 84% of the bucket space is occupied.

References

[1] Baeza-Yates, R. A. and Larson, P. "A Dynamic Storage Allocation Algorithm Suitable for File Structures." *Information Systems*, vol. 15, no. 5, 1990, pp. 515–521.

[2] Baeza-Yates, R. A. "Some Average Measures in m-ary Search Trees." *Information Processing Letters*, vol. 25, no. 6, July 1987, pp. 375–381.

[3] de la Briandais, R. "File Searching Using Variable Length Keys." *AFIPS Western Joint Computer Conference*, San Francisco, CA, March 1959, pp. 295–298.

[4] Fagin, R., Nievergelt, J., Pippenger, N. and Strong, H. R. "Extendible Hashing — A Fast Access Method for Dynamic Files." *ACM Transactions on Database Systems*, vol. 4, no. 3, Sept. 1979, pp. 315–344.

[5] Frederickson, G. N. "Improving Storage Utilization in Balanced Trees." *17th Annual Allerton Conference on Communication, Control and Computing*, Monticello, IL, Oct. 1979, pp. 255–264.

[6] Fredkin, E. "Trie Memory." *Communications of the ACM*, vol. 3, no. 9, Sept. 1960, pp. 490–499.

[7] Gonnet, G. H. and Baeza-Yates, R. *Handbook of Algorithms and Data Structures in Pascal and C*, 2nd edition. Addison-Wesley, Reading, MA, 1991.

[8] Knuth, D. E. *The Art of Computer Programming*, vol. 3: Sorting and Searching. Addison-Wesley, Reading, MA, 1973.

[9] Lew, W. and Mahmoud, H. M. "The Joint Distribution of Elastic Buckets in a Data Structure." Submitted.

[10] Lomet, D. "Partial Expansions for File Organizations with an Index." *ACM Transactions of Database Systems*, vol. 12, no. 1, March 1987, pp. 65–84.

[11] Mahmoud, H. M. *Evolution of Random Search Trees*, John Wiley & Sons, New York, NY, 1992.

[12] Szpankowski, W. "Some Results on V-ary Asymmetric Tries." *Journal of Algorithms*, vol. 9, no. 2, June 1988, pp. 224–244.

Incorporating Dependent and Multimode Failures into Reliability Evaluation of Extra Stage Shuffle-Exchange MINs

Jerry L. Trahan, Daniel X. Wang, and Suresh Rai
Department of Electrical and Computer Engineering
Louisiana State University, Baton Rouge, LA 70803

Abstract

Multistage Interconnection Networks (MINs) provide a good communication medium between multiple processors and memory modules. Previous reliability evaluation efforts for MINs assumed that all failures are statistically independent and that no degraded operational modes exist, though these assumptions are inconsistent with realistic conditions. In this paper, we relax both assumptions and provide efficient algorithms for terminal, broadcast, and K-terminal reliability evaluation in the Shuffle-Exchange Network with an extra stage (SENE), a redundant path MIN. The shock model is used in a modified form to incorporate failure dependency and multiple operational modes into the reliability evaluation. For K-terminal reliability, let $k = |K|$. For an $N \times N$ SENE, the algorithms run in time $O(\log N)$, $O(\log N)$, and $O(k \log N)$, respectively.

1. Introduction

Parallel computers have developed very rapidly in response to the need for high speed computing in many applications. Several parallel computers, such as IBM's RP3 (Hsu *et al.*, 1987; Wang *et al.*, 1989), the University of Illinois' Cedar (Konicek *et al.*, 1991), and Purdue University's PASM (Schwederski *et al.*, 1991), employ a multistage interconnection network (MIN) to provide a communication medium between processors and processors or shared memory modules. A MIN consists of N inputs and N outputs and (typically) n stages of switching elements (SEs), where $n = \log_2 N$. An SE is generally a 2×2 crossbar network and provides either a straight (T-mode) or cross (X-mode) connection. The SEs in one stage are connected to the SEs in adjacent stages by links that are arranged in patterns. Based on these patterns, various MINs are called as Omega, Flip, Indirect binary cube, Modified data manipulator, baseline, and reverse baseline networks. All these networks are topologically equivalent (Bermond *et al.*, 1989). To improve the fault tolerance of the Omega network, an extra stage may be added to provide a redundant path from each input to each output. Thus, if one path fails due to faulty links and/or SEs, an input can still reach an output through another path. This type of Omega network is called the *Shuffle Exchange Network with an Extra stage* (*SENE*).

This work was supported in part by the Air Force Office of Scientific Research under grant AFOSR-91-0025.

K-terminal reliability is defined as the probability that a set of working paths exists from one specified input to each of a specified set K of outputs. For an $N \times N$ MIN and $|K| = 1$ (N), it is also known as *terminal* (*broadcast*) *reliability*.

Previous reliability evaluation algorithms for MINs assumed 2-mode fault models (working or failed) for each component and statistically independent failures (Botting *et al.*, 1989; Varma and Raghavendra, 1989; Cheng and Ibe, 1992; Trahan and Rai, 1992). These assumptions are common in reliability evaluation, as the evaluation problems are intractable for general networks (Colbourn, 1987). These assumptions, however, fail to adequately model real-world situations. For example, Davis *et al.* (1985) and Schwederski *et al.* (1991) discussed instances of dependent failures, or fault side-effects, in the PASM. The 2-mode model leads to an underestimate of the reliability because it does not allow a degraded operational mode of SEs. The assumption of independent failures leads to an overestimate of reliability. Recently, some researchers have addressed the problem of incorporating dependent failure into reliability computations for general networks (Boyles and Samaniego, 1984; Lam and Li, 1986; Le and Li, 1989).

In this paper, we present efficient algorithms for terminal, broadcast and K-terminal reliability evaluation of an SENE composed of identical SEs, allowing multimode and dependent failures. The algorithms assume a 4-mode model of an SE: a fully operational mode, two degraded operational modes, namely stuck-at-T mode and the stuck-at-X mode, and a completely failed mode. Moreover, we assume that links are reliable. We modify the shock model of Boyles and Samaniego (1984) to incorporate dependent failures into the reliability evaluation algorithms of Trahan and Rai (1992). For K-terminal reliability, let $k = |K|$. For an $N \times N$ SENE, the algorithms run in time $O(\log N)$, $O(\log N)$, and $O(k \log N)$, respectively.

The layout of the paper is as follows. Section 2 describes MIN basics. Section 3 discusses the shock model and explains its application to SENE reliability analysis. Section 4 presents results of the algorithms for terminal, broadcast, and K-terminal reliability of SENE and outlines the techniques used to develop them. For brevity, we present the results only. The derivations and proofs are discussed in (Wang, 1992).

2. Background

The Shuffle-Exchange Network with an Extra stage (SENE) (see figure in Cheng and Ibe (1992)) with N inputs and N outputs is defined to be a MIN with $\log_2 N+1$ stages of 2×2 SEs. The outputs of SEs in stage i connect to the inputs of SEs in stage $i+1$ by a shuffle connection, for $i = 0, 1, \ldots, n-1$. Let $\mathrm{SE}_{i,j}$ denote the jth SE in stage i, where $0 \leq i \leq n$ and $0 \leq j \leq N/2-1$. To help develop a fault model, we consider an SE to be composed of two input controllers (ICs), two output controllers (OCs), and a central controller (CC). Figure 1 illustrates the structure of an SE. Paths through the network are established by a path request mechanism in which a CC decides whether the IC should transfer the input to the OC in T- or

X- connection mode.

3. Shock Model and Dependency Analysis

To analyze SE failure dependency, we use a shock model (Boyles and Samaniego, 1984). The shock model and the event based reliability model (EBRM) proposed by Lam and Li (1986) are identical. The shock model was defined for 2-mode components. We generalize its application to include multimode SEs in our reliability analysis. An extension of the EBRM to multimode components, the cause based multimode model (CBMM), does exist (Le and Li, 1989). Our model, however, is more simply defined as an extension of the shock model or EBRM than as a restricted application of the CBMM. Further, we can explain our generalized model in terms of shocks to subcomponents of SEs. Section 3.1 explains the concepts of the model while its application to SENE analysis is described in Section 3.2.

3.1. SHOCK MODEL

The shock model assumes that statistically independent *shocks*, which occur with known probability, cause the failure of network components. When a shock occurs, it causes the failure of a specific component or set of components. We say that the component or components are *affected* by the shock. A shock affecting a single component is called an *individual shock* (IS), while a shock affecting multiple components is called an *external shock* (ES).

For a network with n components, up to 2^n-1 shocks may be defined theoretically. Most shocks, however, may never happen in the real world. Therefore, we need to consider only those shocks whose occurrence is reasonable in the specified real world conditions.

3.2. SENE FAILURE ANALYSIS

To apply the shock model to a SENE in which SEs may operate in degraded modes, we will expand the notion of an IS, while leaving the notion of an ES unchanged. Instead of associating a single IS with a single component, we will associate one IS for each failed or degraded working mode of a component. Technically, a CC having a stuck-at logic fault can produce a degraded working mode for an SE. Class 1 shocks (defined below) model this scenario.

For ESs, we restrict our analysis to two modes only, so the occurrence of an ES causes all affected SEs to fail. SEs that are far apart are unlikely to be affected by a single ES. In general, the classes of ESs that we define are motivated by the failure of one SE causing the failure of other SEs due to the links connecting them. For shared-memory computers, each processor reads from or writes to the shared-memory through the MINs and so communication flows in both directions. When the forward (reverse) part of an SE is failed, this SE will send garbage messages to either one or both of the SEs in the next (previous) stage that are connected to it and may cause one or both of them to fail. Hence, we assume that an external shock causes a failure in adjacent SEs either in the forward direction (towards the network outputs) or in the backward direction (from fault to network inputs). In a practical design, a failure of an output (input) controller can create a forward (backward) external shock. Such dependent failures have been noted in MINs by Davis *et al.* (1985) and Schwederski *et al.* (1991). In the terminology of Schwederski *et al.*, the shocks that we have described above correspond to fault side-effects with forward reach of 1, backward reach of 1, and span of 2. Class 2, 3out, and 3in shocks (defined below) model this scenario.

To help compute the reliabilities, we consider the following *classes* of shocks. Each class of shock will affect a certain structured set of SEs. For example, a Class 3out shock (defined below) will affect an SE and the two SEs to which it is connected in the next stage. We define such a shock for every SE. Because the structures affected by the same class of shocks are the same and all SEs are identical, the probabilities that the same class of shock occur are identical for all such shocks. The probability that each class of shock occurs may be time dependent. The purpose of our work is to find out a relationship between the probabilities of each class of shock and the reliability of the whole network.

Class 1 shock. Exactly one SE is damaged or fails.

A Class 1 shock is an IS that affects only one SE. We modify the shock model as described above to handle stuck-at-T and stuck-at-X modes of SEs. Let $Z_{ij}(1,f)$ denote the shock that affects SE_{ij} to be completely failed, let $Z_{ij}(1,t)$ denote the shock that affects SE_{ij} to be stuck at T mode, and let $Z_{ij}(1,x)$ denote the shock that affects SE_{ij} to be stuck at X mode. The probabilities that the three Class 1 shocks occur are p_f, p_t, and p_x, respectively. Let p_w denote the probability that the three Class 1 shocks affecting SE_{ij} do not occur, hence, $p_w = 1-(p_f+p_t+p_x)$.

Class 2 shock. Exactly two SEs fail simultaneously.

A Class 2 shock is an ES that affects two SEs connected by a link. Four Class 2 shocks affect each SE SE_{ij}, if SE_{ij} is not in the input or output stages. These four shocks will be denoted as $Z_{ij}(2,k)$, where $k = 1, 2, 3, 4$ (Figure 2(a)). If SE_{ij} is in the input stage, then only $Z_{ij}(2,3)$ and $Z_{ij}(2,4)$ affect it. If SE_{ij} is in the output stage, then only $Z_{ij}(2,1)$ and $Z_{ij}(2,2)$ affect it. Let the probability that a Class 2 shock occurs be p_2 and does not occur be $q_2 = 1-p_2$.

Class 3out shock and Class 3in shock. Exactly three SEs fail simultaneously.

Class 3out shock.

A Class 3out shock is an ES that affects an SE and the two SEs to which it is connected by its output links. Each SE in a SENE is affected by three Class 3out shocks denoted as $Z_{ij}(3,k)$, where $k = 1, 2, 3$, except for SEs in the input and output stages (Figure 2(b)). If SE_{ij} is in the input stage, then only $Z_{ij}(3,3)$ affects it. If SE_{ij} is in the output stage, then only $Z_{ij}(3,1)$ and $Z_{ij}(3,2)$ affect it. Let the probability that a Class 3out shock occurs be p_{3o} and does not occur be $q_{3o}= 1-p_{3o}$.

Class 3in shock.

A Class 3in shock is an ES that affects an SE and the two SEs to which it is connected by its input links. There are obviously three Class 3in shocks affecting each SE, denoted as $Z_{ij}(3,k)$, where $k = 4, 5, 6$, except for SEs in the input and output stages (Figure 2(c)). SE_{ij} is affected by only $Z_{ij}(3,5)$ and $Z_{ij}(3,6)$ if it is in the input stage and is affected by only $Z_{ij}(3,4)$ if it is in the output stage. Let the probability that a Class 3in shock is up be p_{3i} and down be $q_{3i} = 1-p_{3i}$.

4. Reliability Evaluation

We now present the results for terminal (TR), broadcast (BR), and K-terminal (KR) reliability measures. Trahan and Rai (1992) developed a straightforward algorithm for TR and recursive algorithms for BR and KR of a SENE under assumptions of independent and 2-mode SE failures. They noted that SENE paths form a simple series-parallel graph for TR and a pair of intersecting binary trees for BR and KR. To incorporate dependent and multimode failures, we follow their concept, but must include a careful and much more detailed accounting of the shocks that may affect the SEs on the paths. Note that a single ES may affect one, two, or three SEs on the relevant paths.

To compute TR, note the following.

a. For a SENE, there exist two paths from each input to each output. These two paths share an SE in the input stage and an SE in the output stage, but are otherwise disjoint.

b. A routing tag can be used to set the connections in switches on a path from an input s to an output d. If an input number is $s = s_1 s_2 \ldots s_n$ and an output number is $d = d_1 d_2 \ldots d_n$, where $s_1, s_2, \ldots, s_n$ and $d_1, d_2, \ldots, d_n$ are the bits of the binary representation of s and d respectively, then the routing tag from s to d is $r = r_1 r_2 \ldots r_n$, where $r_1, r_2, \ldots, r_n$ are given by $r_i = s_i \oplus d_i$, for each $1 \le i \le n$.

Theorem 1. For an input s and an output d in an $N \times N$ SENE, the terminal reliability can be computed using the equation below in $O(\log N)$ time.

$$\begin{aligned} TR(N,s,d) &= (p_t + p_w)^{n+1-\sum_{i=1}^{n} r_i} (p_x + p_w)^{\sum_{i=1}^{n} r_i} q_2^{3n} (q_{3o} q_{3i})^{2n} \\ &+ (p_t + p_w)^{n-1+r_n-\sum_{i=1}^{n-1} r_i} (p_x + p_w)^{2-r_n+\sum_{i=1}^{n-1} r_i} q_2^{3n} (q_{3o} q_{3i})^{2n} \\ &- p_w^2 \left[(p_t + p_w)^{n-1-\sum_{i=1}^{n-1} r_i} (p_x + p_w)^{\sum_{i=1}^{n-1} r_i} \right]^2 q_2^{6n-4} (q_{3o} q_{3i})^{4(n-1)} \end{aligned}$$

Based on the structure of the broadcast paths in the SENE (Figure 3), BR can be computed by a recurrence algorithm. This algorithm is described by Theorem 2.

Theorem 2. For an input *s* in an *N*×*N* SENE, the broadcast reliability can be evaluated using the equation below in *O*(log *N*) time.

$$BR(N) = ((p_t + p_x)p_w^{-1} - 2p_{3o}q_{3o}^{-1})q_2^{-1}\beta^N + \beta q_{3i}q_{3o}^{-1}BR'(N),$$

where $\alpha = p_w(q_2)^3(q_{3o}q_{3i})^2$, $\beta = p_w q_2^2 q_{3o}q_{3i}$, and $BR'(N)$ can be computed by

$$BR'(N) = \alpha^2 q_{3o}^{-2}[BR'(N/2)]^2 + 2(p_t + p_x)p_w^{-1}\alpha\beta^{N/2}BR'(N/2)$$
$$+ 2p_w^{-2}q_{3o}[p_w(p_f + \alpha^{-1}p_w - 1) + p_t p_x q_{3o}]\beta^N.$$

The base case $BR'(4)$ is defined as

$$BR'(4) = [(1-p_f)^2 + 2(p_t p_x + p_t + p_x - p_w)(1-p_f)$$
$$+ 2p_w(q_2 q_{3o})^{-1}]p_w^2 q_2^6 q_{3o}^4 q_{3i}^2.$$

Definition. For K-terminal reliability, we describe each SE on a path from a specified input *s* to an output in *K* as *marked.*

Theorem 3. For an input *s* and a set *K* of outputs in an *N*×*N* SENE, where *k* = |*K*|, the K-terminal reliability can be computed by the equation below in *O*(*k* log *N*) time.

$$KR(K,N,s) = \left[(p_t + p_x - 2p_w p_{3o}q_{3o}^{-1})KR_1(K,N,s) + p_w q_{3o}^{-1}KR_2(K,N,s)\right]q_2^2 q_{3o} q_{3i}^2.$$

Values for $KR_1(K,N,s)$ and $KR_2(K,N,s)$ can be computed by recurrence expressions according to the different cases enumerated below. The base case for *N*=4 can be computed by considering several different cases. For the sake of brevity, the results for the base case are not enumerated.

Case I: Only one child of considered SEs is marked.

1. The T mode in these SEs allow working paths from input *s* to the *k* outputs, then $KR_1(K,N,s)$ and $KR_2(K,N,s)$ can be computed by

$$KR_1(K,N,s) = (p_t + p_w)q_2^3(q_{3o}q_{3i})^2 KR_1\left(K,\frac{N}{2},s\right)$$

$$KR_2(K,N,s) = \left[(p_t + p_w)^2 KR_2\left(K,\frac{N}{2},s\right) + 2(p_t + p_w)(p_x + p_f + \alpha^{-1}p_w - 1)KR_1\left(K,\frac{N}{2},s\right)\right]$$
$$\cdot q_2^6(q_{3o}q_{3i})^4.$$

2. The X mode in these SEs allow working paths from input *s* to the *k* outputs, then $KR_1(K,N,s)$ and $KR_2(K,N,s)$ can be computed by the equations above, exchanging p_t and p_x.

Case II: Both children of considered SEs are marked. Let K_l and K_r be the subsets of *K* that lie in the left and right subgraphs, respectively.

$KR_1(K,N,s)$ can be computed by

$$KR_1(K,N,s) = KR_1\left(K_l,\frac{N}{2},s\right)KR_1\left(K_r,\frac{N}{2},s\right)\cdot p_w q_2^3 q_{3o} q_{3i}^2.$$

$KR_2(K,N,s)$ can be computed by

1. If input *s* can access the right (left) subgraph if considered SEs are set in T (X) mode, then

$$
\begin{aligned}
KR_2(K,N,s) = \Big[& p_w^2 q_{3o}^{-2} KR_2\left(K_l,\frac{N}{2},s\right) KR_2\left(K_r,\frac{N}{2},s\right) \\
& + 2 p_t p_w q_{3o}^{-1} KR_1\left(K_l,\frac{N}{2},s\right) KR_2\left(K_r,\frac{N}{2},s\right) \\
& + 2 p_x p_w q_{3o}^{-1} KR_2\left(K_l,\frac{N}{2},s\right) KR_1\left(K_r,\frac{N}{2},s\right) \\
& + 2\left(p_w\left(p_f + \alpha^{-1} p_w - 1\right) q_{3o}^{-1} + p_t p_x\right) KR_1\left(K_l,\frac{N}{2},s\right) KR_1\left(K_r,\frac{N}{2},s\right)\Big] \\
& \cdot q_2^6 q_{3o}^4 q_{3i}^4 .
\end{aligned}
$$

2. If input s can access the left (right) subgraph if considered SEs are set in T (X) mode, then KR_2 can be computed by the equation above, exchanging p_t and p_x.

References

J. C. Bermond, J. M. Fourneau, and A. Jean-Marie (1989), "A graph theoretical approach to equivalence of multistage interconnection networks," *Discr. Appl. Math.*, vol. 22, pp. 201-214.

C. Botting, S. Rai and D. P. Agrawal (1989), "Reliability computation of multistage interconnection networks," *IEEE Trans. Reliability,* vol. 38, no. 1, pp. 138-145.

R. A. Boyles and F. J. Samaniego (1984), "Modeling and inference for multivariate binary data with positive dependence," *J. Am Stat. Assoc.*, vol. 79, pp. 188-193.

X. Cheng and O. C. Ibe (1992), "Reliability of a class of multistage interconnection networks," *IEEE Trans. Par. Distr. Comput.*, vol. 3, no. 2, pp. 241-246.

C. Colbourn (1987), *The Combinatorics of Network Reliability*, Oxford Univ. Press, New York.

N. J. Davis IV, W. T.-Y. Hsu, and H. J. Siegel (1985), "Fault location techniques for distributed control interconnection networks," *IEEE Trans. Comput.*, vol. C-34, pp. 902-910.

Y. Hsu, W. Kleinfelder, and C. J. Tan (1987), "Design of a combining network for the RP3 project," *Proc. of Int'l Conf. on VLSI Technology, Systems, and Applications,* pp. 349-353.

J. Konicek *et al.* (1991), "The organization of the CEDAR system," *Proc. Int'l Conf. on Parallel Processing,* pp. I_49-I_56.

Y. F. Lam and V. O. K. Li (1986), "Reliability modeling and analysis of communication networks with dependent failures," *IEEE Trans. Commun.*, vol. COM-34, no. 1, pp. 82-84.

K. V. Le and V. O. K. Li (1989), "Modeling and analysis of systems with multimode components and dependent failures," *IEEE Trans. Reliability,* vol. 38, no. 1, pp. 68-75.

T. Schwederski, E. Bernath, G. Roos, W. G. Nation, and H. J. Siegel (1991), "Fault side-effects in fault tolerant multistage interconnection networks," *Proc. of Int'l Conf. on Parallel Processing,* pp. I_313-I_317.

J.L. Trahan and S. Rai (1992), "Reliability evaluation of extra stage shuffle-exchange MINs," to appear in *Proc. 1992 Int'l. Conf. Computer Communication*, Genova, Italy.

A. Varma and C. S. Raghavendra (1989), "Reliability analysis of redundant-path interconnection networks," *IEEE Trans. Reliability*, vol. 38, no. 1, pp. 130-137.

D. X. Wang (1992), "Incorporating dependent and multimode failures into reliability evaluation of multistage interconnection networks," M.S. thesis, Louisiana State Univ.

S. Wang, Y. Hsu and C.J. Tan (1989), "A high performance VLSI message switch for multiprocessor systems," Research Report No. 54874, IBM Research Division.

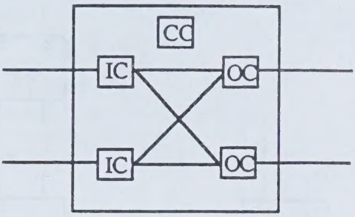


Figure 1. Structure of an SE.
IC: input controller, OC: output controller
CC: central controller

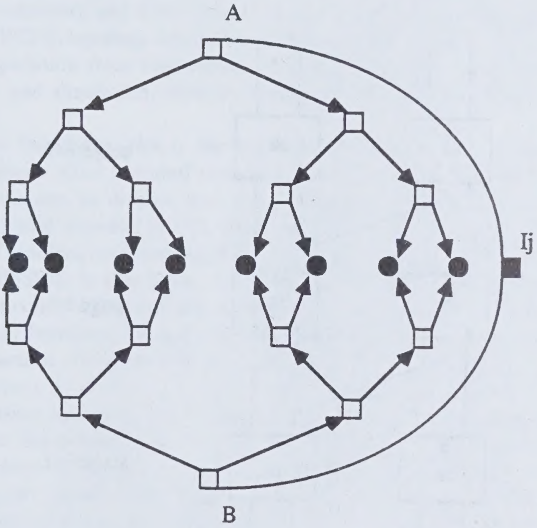
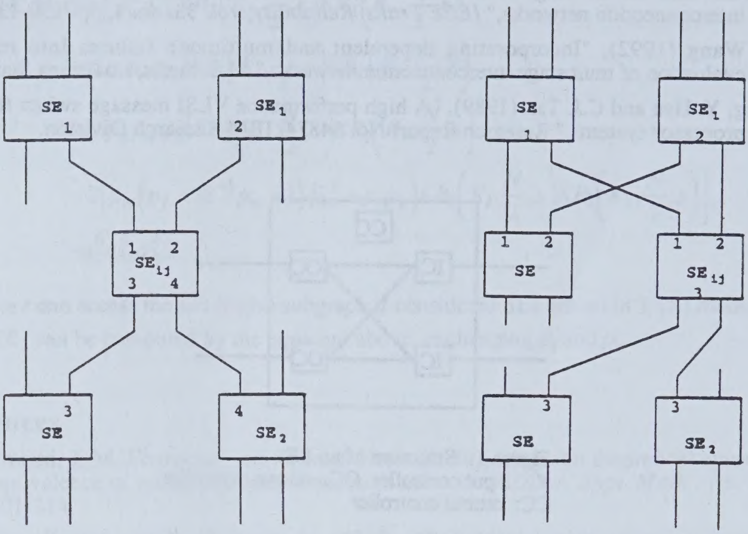
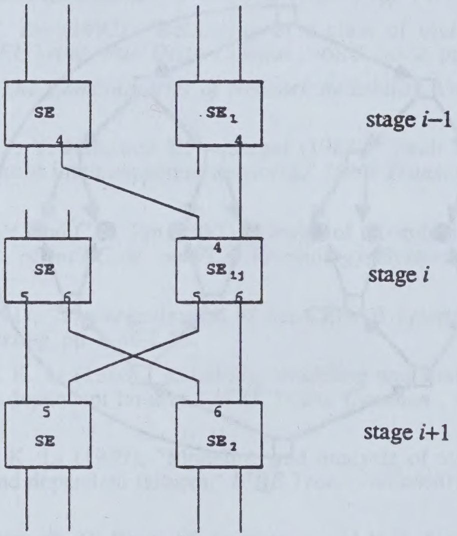


Figure 3. Broadcast paths in SENE
● denotes an output stage SE


SE
1
SE₁
2
1
2
SE_ij
3
4
3
SE
4
SE₂
SE
1
SE₁
2
1
2
SE
1
2
SE_ij
3
3
SE
3
SE₂

(a) Class 2 shock (b) Class 3out shock


SE
SE₁
4
4
stage i–1
SE
4
SE_ij
stage i
5
6
5
6
5
SE
6
SE₂
stage i+1

(c) Class 3in shock

Figure 2. Classes of shocks

On the Power of Discontinuous Approximate Computations [1]

Karl Aberer
International Computer Science Institute
1947 Center Street, Suite 600, Berkeley, CA 94704
email: aberer@icsi.berkeley.edu

Bruno Codenotti
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche, Pisa, Italy
email: codenotti@iei.pi.cnr.it

Abstract. Comparision operations are used in algebraic computations to avoid degeneracies, but are also used in numerical computations to avoid huge roundoff errors. On the other hand, the classes of algorithms using only arithmetic operations are the most studied in complexity theory, and are used, e.g., to obtain fast parallel algorithms for numerical problems. In this paper, we study, by using a simulation argument, the relative power of different sets of operations for computing with approximations. We prove that comparisions can be simulated efficiently and with the same error bounds for most inputs by arithmetic operations when divisions are present. To develop our simulation strategy we combine notions imported from approximation theory and topology with complexity and error bounds.

1 Introduction

In this paper we show that approximate computations over the reals do not suffer very much from the lack of conditional statements, provided that division is allowed. On the other hand we show that the set of operations without division is very poor, if one wants to achieve the goal of keeping the roundoff error small.

The problem studied in this paper has relations with several different research fields, and we attack it by using techniques and ideas related , e.g., to algebraic complexity tools used to remove degeneracies [EC91], topology notions used by numerical analysts to evaluate the distance of a path in a computation from degenerate (ill-conditioned) regions [SS91], and techniques, like arithmetization and simulation, used in complexity to find lower bounds on the size of a circuit [BF91, C91].

To the best of our knowledge, this is the first nontrivial result on the relative power of different sets of operations when roundoff errors are present. Previous work on the complexity of numerical problems can be divided into three categories: complexity results for obtaining ε-approximations without roundoff [CF89, CL91, MST89, TWW88, PR85], complexity bounds obtained in terms of problem conditioning [SS91, S90], and development of theories of computation over the reals [BSS89]. In this paper, we embed the topological notions needed to evaluate the effect of problem conditioning and the approximation notions needed to find bounds on the degree of the rational functions (or polynomials) involved in the simulation process onto the framework of the classical circuit complexity measures (size, depth, width).

The set of operations $S_1 = \{+,-,*,/,>\}$ is used in algebraic computations to avoid degeneracies (i.e. division by zero), but is also used in numerical computations to avoid huge roundoff errors. On the other hand, the class of algorithms using operations from the set $S_2 = \{+,-,*,/\}$ or from the set $S_3 = \{+,-,*\}$ is extremely important, and is the one actually used, e.g., to obtain fast parallel algorithms for numerical problems. In this paper, we study, by using a simulation argument, the relative power of the sets S_1, S_2, and S_3. We prove that S_2 does very efficiently simulate S_1, while S_3 does not; this fact shows and measures the crucial role of division in computations introducing roundoff errors. We also show how to construct algorithms using operations $\{+,-,*,/\}$ which achieve for most inputs the same bounds on the roundoff error as algorithms using operations $\{+,-,*,/,>\}$.

Here we adopt a model of computation which essentially consists of a "layered version" of an arithmetic network [G86], which is simpler to analyze. Arithmetic network and branching programs are generally analyzed (see, e.g., [S90]) to measure the extra cost necessary to avoid degeneracies, e.g., divisions by zero. As already mentioned, our goals are more general, since we view branching instructions (conditionals) as a tool to avoid huge errors in a computation, and we study the conditions under which adaptive computations, i.e. those allowing branching instructions, can be simulated by oblivious ones, i.e. straight-line programs.

---

[1] The work of Bruno Codenotti was partially supported by the Italian National Research Council under the "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 2" and was done while he was at the International Computer Science Institute, Berkeley.

The main contributions of this paper are expressed by the results of section 3. The parameters we simultaneously deal with are the length of the input n, the degree of the approximating functions m, the maximum value M an intermediate result can achieve, the local approximation error ε_{loc}, the global approximation error ε_{glob}, the number k of polynomials defining degenerate regions, which we call *decision polynomials*, the maximum degree d of the above polynomials, and the quantity δ which measures the extension of the degenerate regions. The basic bound, that allows us to find several tradeoffs among the above parameters, is shown in Lemma 8, where it is proved that

$$f(V_\delta) \leq 12kndMe^{-\sqrt{m}}\varepsilon_{loc}^{-1},$$

where $f(V_\delta)$ denotes the intersection of the unit ball with the volume of the degenerate regions. Using the above bound together with other results of section 3, we prove that algorithms using operations from the set S_2 can carry out a simulation by using, under reasonable conditions, rational functions of degree at most $O(\log^2 n)$. On the contrary, we show that algorithms using operations in S_3 require the use of polynomials of degree at least $n^{O(1)}$ in the input size. This gives a measure of the importance of division.

This paper is organized as follows. Section 2 contains the description of the models of computation adopted, the definition of adversary which is used to find bounds on the error, some preliminary results on the error introduced by oblivious computations and some theorems which we import from approximation theory and topology. Section 3 contains the description of our simulation strategy, which gives upper and lower bounds on the cost of avoiding conditionals. In section 4 we report some concluding remarks and open problems.

2 Preliminaries

2.1 Models of Computation

The models of computations adopted here are a slight modification of arithmetic circuits and networks, and branching programs. More precisely, we adopt models which are a *structured* version of the above models, which turn out to be particularly useful for a detailed analysis (see [AC92]). Before describing the models of computation, it is convenient to define which kind of algorithms we consider.

An algorithm using operations $\{+,-,*,/\}$ is called *oblivious rational.* An algorithm using operations $\{+,-,*\}$ is called *oblivious polynomial.* An algorithm using operations $\{+,-,*,/,>\}$ is called *adaptive rational.* An algorithm using operations $\{+,-,*,>\}$ is called *adaptive polynomial.*

The simplest model consists of an arithmetic circuit, with gates $\{+,-,*,/\}$ or $\{+,-,*\}$ fed by a routing network, with gates $\{>\}$ (see Figure 1a). The input data are fed to the routing network which can compute any permutation of these data, and then provide the arithmetic circuit with the chosen permutation. The routing network is a network of comparators (see Figure 1b). We call *arithmetic stage* the part of the computation carried out by the arithmetic circuit, and *combinatorial stage* the part of the computation carried out by the routing network. A slight modification of this model consists of a cascade of "modules" of the previous kind, as shown in Figure 1c. In this case adaptivity can also be used to decide upon partial results. The relations of this model to arithmetic network are obvious (see [G86]). The cost of a computation can be analyzed by using the natural measures of circuit complexity (size, depth, and width). We will also analyze the error that is introduced in the arithmetic part by means of a "layered"

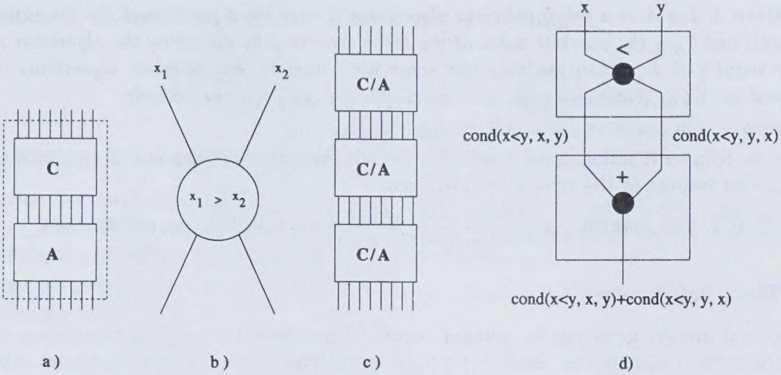


Figure 1: a) A combinatory (C) and arithmetic (A) stage.
b) The combinatory stage consists of comparators.
c) A cascade of modules. C/A is a module of the type introduced in a).
d) Translating a circuit into an expression.

error analysis, in which for any level i of the circuit, we evaluate the maximum attainable error (see section 2).

It is easy to see that any arithmetic circuit corresponds in a straightforward way to an arithmetic expression. On the other hand, by taking into account parentheses, an arithmetic expression uniquely defines a computation, which can be easily interpreted as the evaluation of an arithmetic circuit. To describe a routing network as an expression, it is sufficient to introduce *conditional functions* of the form

$$\operatorname{cond}(P, e_1, e_2) = \begin{cases} e_1, & \text{if } P \text{ true,} \\ e_2, & \text{otherwise,} \end{cases} \tag{1}$$

where e_1 and e_2 are expressions built up from arithmetic and conditional functions and P is a primitive predicate depending on one or more input data, e.g., $P(x) \equiv (x > 0)$ or $P(x, y) \equiv (x > y)$.

A numerical computation (consisting of one combinatorial and one arithmetic stage) taking k input values and computing m output values, is described by the expressions $e_1(x_1, \ldots, x_k), \ldots, e_m(x_1, \ldots, x_k)$. The combinatorial stage consists of evaluating all conditional functions according to the inputs and the arithmetic stage consists of evaluating all arithmetic functions according to the input values. Several combinatorial/arithmetic stages can be combined by composing the corresponding expressions. The relation of this model to the circuit-model is illustrated in Figure 1d.

In this model, the cost of a computation can be evaluated according to the maximum depth of all the expressions in the stage, the maximum number of different subexpressions at one level, and the overall size of the expressions.

2.2 Errors and Adversaries

It is very difficult to express lower bounds on the error an algorithm produces, because the notion of error is heavily instant-dependent. Typically, a lower bound which has to hold for all instances should be zero. Therefore we adopt the following strategy, which is based on an adversary argument (a similar approach can be found in [CLR91], or [TWW88]). In other words we assume an adversary can decide about the input distribution. This leads to the following definition of *unavoidable roundoff error.*

Definition 1 *Let $\mathcal{A}$ be a set of oblivious algorithms for solving a problem, I the set of problem instances, and $\varepsilon_{A(i)}$ the absolute value of the error occurring by executing the algorithm $A \in \mathcal{A}$ on the input $i \in I$. Then the roundoff error for oblivious and adaptive algorithms can be expressed as $\min_{A\in\mathcal{A}}(\max_{i\in I} \varepsilon_{A(i)})$, and $\max_{i\in I}(\min_{A\in\mathcal{A}} \varepsilon_{A(i)})$, respectively.*

Theorem 1 $\min_{A\in\mathcal{A}}(\max_{i\in I} \varepsilon_{A(i)}) \geq \max_{i\in I}(\min_{A\in\mathcal{A}} \varepsilon_{A(i)})$.

We now define a function which quantifies the relative power of adaptive and oblivious algorithms, with respect to the error they introduce.

Definition 2 *The quantity* $\log \dfrac{\max_{i\in I}(\min_{A\in\mathcal{A}} \varepsilon_{A(i)})}{\min_{A\in\mathcal{A}}(\max_{i\in I} \varepsilon_{A(i)})}$ *is called the pay-off function.*

2.3 Roundoff Errors

A numerical process gives rise to roundoff errors. These errors depend on the system chosen to represent the real numbers. Since our goal is to analyze roundoff errors, we do not take into account the question of under/overflow, and we may assume that a countable subset R of the real numbers, e.g., floating point numbers with arbitrary integer exponents, rational numbers or fixed point numbers of arbitrary size, is given. Given $R \subseteq \mathbb{R}$ we describe the roundoff error by means of a rounding function $r : \mathbb{R} \to R$ with the properties

$$r(x) = x + \varepsilon_R(x), \ |\varepsilon_R(x)| \leq h, \tag{2}$$

where h is a constant depending only on R. Such a rounding function exists, for the representation systems mentioned above. In the models of computation of section 2.1, the arithmetic stages were essentially straight-line programs built up in terms of primitive functions from a set $F = \{f_1, \dots, f_k\}$. The roundoff errors lead to local errors in the evaluation of these primitive functions. We assume that the primitive functions satisfy a condition of the form

$$f_i(x_1 + \varepsilon_1, \dots, x_n + \varepsilon_n) = f_i(x_1, \dots, x_n) + \varepsilon_1\alpha_1(x_1, \dots, x_n) + \dots + \varepsilon_n\alpha_n(x_1, \dots, x_n). \tag{3}$$

Note that for differentiable primitive functions, as for example addition and multiplication, such a property is naturally satisfied. A numerical operation is then described by the function

$$f_i^r(x_1, \dots, x_n) := r(f_i(x_1, \dots, x_n)), \ x_1, \dots, x_n \in R, \ i = 1, \dots, k.$$

2.4 Layered Error Analysis

In this section, we prove some bounds on the error introduced by oblivious algorithms. We will use these bounds in section 3.

Lemma 1 *Let f be a composition of primitive functions. Given a computation graph of depth t associated to the function f, the "roundoff error" ε_t can be evaluated according to the formula:*

$$\begin{aligned} \varepsilon_i &= \alpha_i\varepsilon_{i-1} + \varepsilon_{R_i}, \ i = 1, \dots, t, \ |\varepsilon_{R_i}| \leq h, \\ \varepsilon_0 &= \begin{cases} \varepsilon_{R_0}, & \textit{if we include the perturbation error,} \\ 0, & \textit{otherwise,} \end{cases} \end{aligned}$$

where the α_i, $i = 1, \dots, t$, are quantities depending on a primitive function and partial results.

Proof. We follow one path in the computation graph from the input to the output nodes. Let t be the length of the path, and let v_{j_i}, $i = 1, \dots, t$, be the sequence of values computed along this path. Then the error is evaluated using (2) and (3)

$$\begin{aligned} \varepsilon_i &= f_i^r(x_1 + e_1, \dots, v_{j_{i-1}} + \varepsilon_{j_{i-1}}, \dots, x_n + e_n) - f_i(x_1, \dots, v_{j_{i-1}} \dots, x_{n_i}) \\ &= e_1\alpha_1 + \dots + \varepsilon_{j_{i-1}}\alpha_{j_i} + \dots + e_{n_i}\alpha_{n_i} + \varepsilon_{R_i} = \varepsilon_{j_{i-1}}(\alpha_{j_i} + \frac{e_1}{\varepsilon_{j_{i-1}}}\alpha_1 + \dots + \frac{e_{n_i}}{\varepsilon_{j_{i-1}}}\alpha_{n_i}) + \varepsilon_{R_i}. \end{aligned}$$

∎

Corollary 1 *Under the assumptions of lemma 1, if* $\varepsilon_0 = 0$, *then*

$$\varepsilon_t = \sum_{i=1}^{t} \varepsilon_{R_i} \prod_{j=i+1}^{t} \alpha_j, \quad \textit{and} \quad |\varepsilon_t| \leq h \sum_{i=1}^{t} \prod_{j=i+1}^{t} |\alpha_j|.$$

2.5 Bounds from Approximation Theory and Topology

In section 3 we will make use of the following lemmas, which we import from approximation theory ([R69, D88]) and topology ([G82, W39]). The first three lemmas provide bounds on the best rational and polynomial approximation to the function $|x|$.

Lemma 2 ([N64]) *Let* $a(x) = |x|$ *and* $r(x)$ *be a best-approximating rational function of degree* m. *Then we have*

$$\frac{e^{-9\sqrt{m}}}{2} < \max_{x \in [-1,1]} |a(x) - r(x)| \leq \begin{cases} 3e^{-\sqrt{m}}, & m \textit{ even} \\ 3e^{-\sqrt{m-1}}, & m \textit{ odd.} \end{cases}$$

Lemma 3 *A function* $r(x)$ *achieving the upper bound of lemma 2 is*

$$r(x) = \frac{x(p(x) - p(-x))}{p(x) + p(-x)}, \quad \textit{where } p(x) = \prod_{j=0}^{m-1} (x + t^j), \; t = e^{\frac{-1}{\sqrt{m}}}.$$

Proof. (Hint.) Follows from [R69] page 128. ∎

Lemma 4 *Let* $p(x)$ *be a polynomial of degree* m *and* $a(x) = |x|$, *and let* K *be a constant. Then we have*

$$\frac{K}{m} < \max_{x \in [-1,1]} |a(x) - p(x)| \leq \frac{2}{(2m+1)\pi}.$$

Proof. (Hint.) The upper bound can be obtained by using Tchebycheff or Legendre polynomials. The lower bound follows from [L52]. ∎

We now report a result on the volumes of tubes from [O85], which we will need to estimate the volumes of the *degenerate regions* created by the approximate simulation process (see section 3).

Theorem 2 *Suppose* M *is a real, purely d-dimensional variety in* $\mathbb{R}^n$. *Suppose further that* M *is the complete intersection of the polynomials* $p_1, \ldots, p_{n-d}$. *Let* $D = \max_{1 \leq i \leq n-d} deg(p_i)$, *and* $f(V_\delta)$ *be the fraction of the volume of the unit ball in* $\mathbb{R}^n$ *which is within distance* $\delta \leq 1$ *of* M. *Then*

$$f(V_\delta) \leq 2(n-d) \sum_{k=n-d}^{n} \binom{n}{k} (2D\delta)^k.$$

3 Main Results

This section contains our main results. We analyze adaptive computations for which the pay-off function does not tend to zero, and we show a simulation procedure which allows us to investigate the relative power of different sets of operations.

We first show how to simulate the expressions cond$(x > 0, y, z)$ by rational functions, and analyze the cost of this process in terms of the degree of the rational functions (i.e. the maximum degrees of the denominator and numerator). The goal of this simulation is to find oblivious algorithms behaving like adaptive ones, for most inputs. We associate to the decisions corresponding to adaptive algorithms suitable semi-algebraic varieties which describe singularities. Analogously, we introduce semi-algebraic varieties which describe the degenerate regions "close"

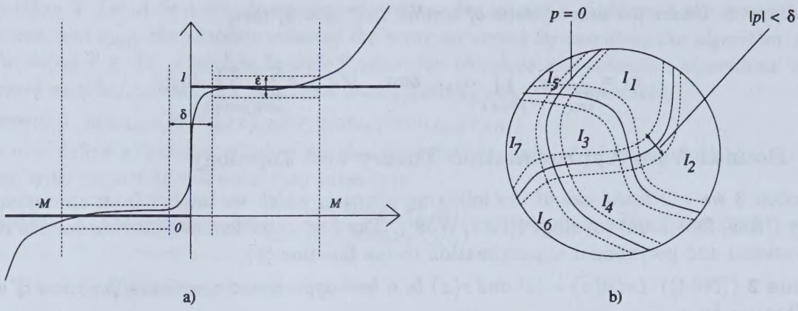


Figure 2: a) Approximation of the step-function.
b) The critical regions.

to singularities. These regions correspond to the inputs for which the approximate simulation cannot take place. We then prove lower and upper bounds on the volumes of such regions, and from these bounds, we are able to estimate the fraction of inputs for which we can approximately simulate adaptive algorithms. We also extend the above results taking care of the effects of roundoff errors, and by using estimates of the error introduced by oblivious algorithms. It turns out that the simulation is feasible at a very reasonable price, provided that it involves rational functions. The restriction to polynomial computations is proved to be very expensive.

3.1 Simulation of Conditionals

In section 2 we introduced the notation cond($a > 0, 1, 0$) to denote expressions like *if* $a > 0$ *then* 1 *else* 0. We replace the expression cond($a > 0, 1, 0$) by an arithmetic circuit (oblivious algorithm) simulating it. From this simulation - to be described below - we immediately get, by arithmetization, a simulation of a general conditional cond($x > 0, y, z$). In fact

$$\text{cond}(x > 0, y, z) = \text{cond}(x > 0, 1, 0) * y - (\text{cond}(x > 0, 1, 0) - 1) * z.$$

Definition 3 *A function* $f : \mathbb{R} \to \mathbb{R}$ *simulates cond*($a > 0, 1, 0$) *iff we have*

$$|f(x) - s(x)| < \varepsilon, \ \forall x \in [-M, -\delta) \cup (\delta, M],$$

where $s(x) = \begin{cases} 1, & \textit{if } x > 0 \\ 0, & \textit{otherwise} \end{cases}$*, and* $M > \delta > 0$ *(see figure 2a).*

We use lemmas 2 and 4 to approximate the function $s(x)$. We do this by observing that,

$$s(x) = \frac{x - |x|}{2x}, \text{ for } x \neq 0.$$

Lemma 5 *Let* $M > 0$. *Then there exists a rational function* $r(x)$ *such that*

$$Me^{-9\sqrt{m}} \le max_{x \in (-M,M)} |r(x) - |x|| \le M\gamma_m, \ \gamma_m = \begin{cases} 3e^{-\sqrt{m}}, & m \textit{ even} \\ 3e^{-\sqrt{m-1}}, & m \textit{ odd} \end{cases}$$

Proof. (Sketch.) Follows from lemma 2 , by using the substitution $r(x) := \frac{1}{M}\tilde{r}(\frac{x}{M})$, where $\tilde{r}$ is the rational function of lemma 2. Note that r and $\tilde{r}$ have the same degree. ∎

Lemma 6 $\frac{e^{-9\sqrt{m}}}{2} \le \max_{x \in [-M,-\delta) \cup (\delta, M]} |r(x) - s(x)| \le \frac{1}{2}\gamma_m M \delta^{-1}$.

If we want the rational approximation to be an ε-approximation we have the following.

Corollary 2 *If* $\delta \ge \frac{1}{2}\gamma_m M \varepsilon^{-1}$ *then* $\max_{x \in [-M,-\delta) \cup (\delta, M]} \mid r(x) - s(x) \mid \le \varepsilon$.

A correspondent lemma could be also proven for polynomial approximations, by using lemma 4. We omit it here for the sake of brevity.

3.2 Approximate Simulation of Adaptive Computations

The above arguments naturally suggest a procedure of *approximate simulation.*

Definition 4 *An algorithm A for computing $g : I \to O$ ε-approximately simulates an algorithm B for computing $g : I \to O$ on a subset $I' \subseteq I$ if there is ε such that $\forall x \in I'$ $|A(x) - B(x)| \leq \varepsilon$.*

Definition 5 *An adaptive algorithm taking n input values defines the semi-algebraic varieties $V = \bigcup_{i=1}^{k} \{x_1, \ldots, x_n : p_i(x_1, \ldots, x_n) = 0\}$, and $V_\delta = \bigcup_{i=1}^{k} \{x_1, \ldots, x_n : |p_i(x_1, \ldots, x_n)| < \delta\}$, where $p_i(x_1, \ldots, x_n)$ is a polynomial for which a decision $p_i(x_1, \ldots, x_n) > 0$ is made during the computation. We call V the critical and V_δ the δ-critical variety of the adaptive algorithm.*

Definition 6 *We denote with $M(r)$ the maximum absolute value of all intermediate quantities created during the computation on inputs $(x_1, \ldots, x_n) \in B(r)$, where $B(r) \in \mathbb{R}^n$ is the ball of radius r centered in the origin.*

Theorem 3 *An adaptive algorithm, associated with V and $M(r)$, can be approximately simulated by an oblivious algorithm for inputs in $I' = V_\delta \cap B(r)$.*

Proof. (Hint.) Use lemma 6 together with the simulation arguments. ∎

We now adapt the theorems on the volume of tubes reported in section 2 to estimate how large degenerate regions (critical varieties) are. We do this, by first finding bounds on the volume of the regions $\{x_1, \ldots, x_n : |p(x_1, \ldots, x_n)| < \delta\}$.

Lemma 7 *Let n be the length of the input and let d be the degree of p. Let $f(V_\delta)$ be the intersection of $B(1)$ with the set $\{x_1, \ldots, x_n : |p(x_1, \ldots, x_n)| < \delta\}$. Then*

$$f(V_\delta) \leq 2 \sum_{i=1}^{n} \binom{n}{i} (2d\delta)^i = 2(1 + 2d\delta)^n - 2.$$

Proof. It follows from the application of theorem 2 to the case of hypersurfaces. ∎

Theorem 4 *Let n be the length of the input and let be d the maximum degree of p_i in V_δ. Let $f(V_\delta)$ be the intersection of $B(1)$ with the set $\bigcup_{i=1}^{k} \{x_1, \ldots, x_n : |p_i(x_1, \ldots, x_n)| < \delta\}$, then*

$$f(V_\delta) \leq 2k \sum_{i=1}^{n} \binom{n}{i} (2d\delta)^i = 2k(1 + 2d\delta)^n - 2k.$$

Corollary 3 *Let $f(V_\delta)$ be as in theorem 4. If $d\delta n < \frac{\tau}{2}$, then $f(V_\delta) < \alpha \leq 1$, for n large enough, where $\tau = \log(\frac{\alpha}{2k} + 1)$.*

Lemma 8 $f(V_\delta) \leq 4kndM\gamma_m\varepsilon_{loc}^{-1}$, *where ε_{loc} denotes the local approximation error.*

Now we can give a lower bound on the degree of the approximating function. We assume that the global error ε_{glob}, as given in definition 4, satisfies $\varepsilon_{glob} = n^c\varepsilon_{loc}$ for large n, where c is a constant.

Lemma 9 *The global error ε_{glob} satisfies $\varepsilon_{glob} < \varepsilon$ for $x \in V_\delta$, with $f(V_\delta) < \alpha$, if*

$$m \geq c \log^2(\log^{-1}(\frac{\alpha}{2k} + 1)ndM\varepsilon^{-1}), \quad \textit{where c is a constant.}$$

Theorem 5 *If the global approximation error ε_{glob}, as given in definition 4 satisfies $\varepsilon_{glob} = n^c\varepsilon_{loc}$, where c is a constant, and if $k, M, d \leq n^{O(1)}$, then the degree m of the rational functions used in the simulation satisfies $m = O(\log^2 n)$.*

This theorem tells us that the simulation is relatively cheap as long as the algorithm we simulate is numerically stable.

3.3 Lower Bounds for Oblivious Algorithms

We give conditions on the data representation and primitive functions (which are satisfied by the systems used in practice) for which the adversary has the power sufficient to produce nontrivial lower bounds on the error.

Condition 1: We assume that an adversary can, for any path in any computation graph, provide input data for which there exists $\tilde{h}$, $|\tilde{h}| < h$, such that, $|\varepsilon_{R_i}| > \tilde{h}$, and $sign(\tilde{h}) = sign(\varepsilon_{R_i})$.

Condition 2: a) There exists a subset $R' \subseteq R$ such that for all $\tilde{y} \in R'$, for all n-ary primitive functions f and all $i \in \{1,\dots,n\}$ there exist arguments $\tilde{x_1},\dots,\tilde{x_n} \in R$ such that $\tilde{x_i} \in R'$, $f^r(\tilde{x_1},\dots,\tilde{x_n}) = \tilde{y}$, $f(\tilde{x_1},\dots,\tilde{x_n}) = y$, $|y-\tilde{y}| \geq \tilde{h} > 0$, and $sign(y-\tilde{y}) = const.$ b) Furthermore for all primitive functions f and $\tilde{y} \in R$ there exist $\tilde{x_1},\dots,\tilde{x_n} \in R$ such that $f^r(\tilde{x_1},\dots,\tilde{x_n}) = \tilde{y}$.

Lemma 10 *If the computation is carried out on a tree, then Condition 2 implies Condition 1.*

Lemma 11 *If Condition 1 is satisfied, then, in the worst case, we have*

$$|\varepsilon_t| \geq \tilde{h} \sum_{i=1}^{t} \prod_{j=i+1}^{t} \alpha_j = \tilde{h}t + \tilde{h} \sum_{i=1}^{t} \beta_i, \text{ where } \varepsilon_{R_i} = \tilde{h} - \gamma_i, \text{ and } \textstyle\prod_{j=1}^{i} \alpha_j = 1 + \beta_i.$$

Proof. We have $\varepsilon_t = \tilde{h}t + \tilde{h}\sum_{i=1}^{t} \beta_i - \sum_{i=1}^{t} \gamma_i \beta_i - \sum_{i=1}^{t} \gamma_i$. The proof follows from the fact that Condition 1 allows the adversary to provide data such that $sign(\gamma_i) \neq sign(\tilde{h})$, $i = 1,\dots,t$. ∎

Corollary 4 *Under the hypothesis of lemma 11, the representation error for oblivious algorithms is in the worst case lower bounded by* $|\tilde{h}|C$ *if* $\alpha_i \geq 1$, $i = 1,\dots,t$, *or by* $|\tilde{h}|(1+\delta)^C$, *if* $\alpha_i \geq 1+\delta$, $\delta > 0$, *where* C *is the longest path in the computation graph of minimum depth.*

Note that the conditions $\alpha_i \geq 1$ or $\alpha_i \geq 1+\delta$ are assumptions on primitive functions and/or intermediate data.

Corollary 5 *Assume that only one primitive function is used, and* $\alpha_i = \alpha > 0$, $i = 1,\dots,t$, *is independent of the intermediate data. Then we have*

$$|\varepsilon_C| \geq |\tilde{h}| \, |\sum_{i=1}^{C} \alpha^i| = \begin{cases} |\tilde{h}| \, |\frac{(\alpha - \alpha^{C+1})}{(1-\alpha)}|, & \alpha \neq 1, \\ |\tilde{h}| \, C, & \alpha = 1. \end{cases}$$

Theorem 6 *Under the hypothesis of lemma 11, let* g *be a differentiable function finitely composed of primitive functions. Let* D *be a lower bound on the minimal depth of a computation graph for the computation of* g*. Then the error introduced by an oblivious algorithm for* g *is in the worst case at least* $\tilde{h}(|g'| + D)$, *if* $\alpha_i \geq 1$ *or* $\tilde{h}(|g'| + \gamma^D)$, $\gamma > 1$, *if* $\alpha_i > 1$.

3.4 From Approximation and Topology to Complexity and Conditioning

In this section, we make a simplifying assumption on the nature of the computations we are considering. It is possible to prove that this is not a loss of generality. We assume that the simulation has the goal of approximating a computation carried out on a circuit (see section 2) consisting only of two layers, one combinatorial stage and one arithmetic stage. Let the roundoff error associated to this computation be such that the payoff function is strictly greater than zero. This means that the error introduced is $c\varepsilon_R$, with $c < 1$, where ε_R is the error unavoidable by oblivious algorithms performing the same computation. The above scenario leads to the following result.

Lemma 12 *Let* D_c *be the depth of the part of the circuit implementing the combinatorial stage. Let* g *be the (differentiable) function to be computed by the overall circuit. Then, we have* $\varepsilon_{loc} < \frac{\varepsilon_R}{|g'|D_c}$.

Proof. The error introduced in the approximation of the combinatorial stage is $\varepsilon_{loc}D_c$, so that the error of the overall computation is $|g'|\varepsilon_{loc}D_c$, where the factor $|g'|$ plays the role of amplifying the error. The proof now follows from the assumption that the payoff function is strictly positive. ∎

We can now state a very important result, concerning the feasibility of the simulation by rational functions for $x \in V_\delta$, in the presence of roundoff errors.

Theorem 7 $\frac{e^{-9\sqrt{m}}}{2} \leq \varepsilon_{loc} < \frac{\varepsilon_R}{|g'|D_c}$.

Proof. Follows from lemma 6 and lemma 12. ∎

From this we can also evaluate the cost of the rational simulation in the presence of roundoff errors in order to achieve a strictly positive payoff function.

Corollary 6 $m > 9\log^2(2\varepsilon_R^{-1}|g'|D_c)$.

If we further assume that an oblivious algorithm achieves the error defined in section 3.3, then we get the bound

$$m > 9\log^2(\frac{2D_c|g'|}{hD_a}),$$

where D_a denotes the minimal depth of an oblivious algorithm for computing g.

Finally, we get the following result for the degree m in the presence of roundoff errors.

Theorem 8 *If g is such that $|g'| \leq n^{O(1)}$, then $m = O(\log^2 n)$.*

Theorem 8 essentially says that the approximate simulation is "cheap" provided that we are not trying to solve an ill-posed problem.

For the lack of space we did not include the corresponding theorems and results for a polynomial simulation, which lead to the conclusion that polynomials of degree at least $n^{O(1)}$ are necessary. Another issue that did not fit into these pages is that of computations over complex numbers. In this case, the results obtained were similar to those presented for computations over the reals, with some additional features related to the fact that the non negligible volumes of tubes over complex varieties lead to non trivial lower bounds [D88, W39].

4 Conclusions and Further Work

In this paper we have shown that approximate computations over the reals do not suffer very much from the lack of conditionals, provided that division is allowed. By simulation, we have proved that it is possible to construct oblivious algorithms which, almost everywhere, behave like adaptive algorithms. The cost of this construction is very low if division is allowed and is very high if division is not allowed. The techniques employed here are very general and we believe will be used in several different areas.

One of the open problems we hope to see solved by using our techniques is related to the development of numerically stable NC algorithms. For example, Gaussian elimination with pivoting is P-complete [V89], whereas Gaussian elimination can be performed in NC. Can the simulation process described in this paper be applied to Gaussian elimination with pivoting, so that we obtain an algorithm almost everywhere reliable and, as the same time, parallelizable? More in general, assume that an adaptive algorithm is P-complete. Is its oblivious counterpart, obtained by approximate simulation, also P-complete?

The continuation of this work will include a detailed analysis of the polynomial approximation process, as well as the analysis carried out for computations over the complex numbers, where the regions of degeneracies have non trivial lower bounds [D88].

References

[AC92] Aberer, K., Codenotti, B. (1992). Towards a Complexity Theory of Approximation. *TR-92-012, International Computer Science Institute.*

[BF91] Babai, L., Fortnow, L., (1991). Arithmetization: A New Method in Structural Complexity Theory. *Computational Complexity, Vol.* **1**, *No. 1, pp 41-67.*

[BSS89] Blum, L., Shub, M., Smale, S. (1989). On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness. Recursive Functions and Universal Machines, *Bulletin of AMS, Vol.* **21**, *No. 1, pp 1-36.*

[C91] Cleve, R., (1991). Towards Optimal Simulation of Formulas by Bounded-width Programs. *Computational Complexity, Vol.* **1**, *No. 1, pp 91-105.*

[CF89] Codenotti, B., Flandoli, F., (1989). A Monte Carlo Method for the Parallel Solution of Linear Systems. *Journal of Complexity, Vol.* **5**, *pp 107-117.*

[CL91] Codenotti, B., Leoncini, M., (1991). Matrix Inversion in RNC^1. *Journal of Complexity, Vol.* **76**, *No. 3, pp 282-295.*

[CLR91] Codenotti, B., Leoncini, M., Resta, E., (1991). Oracle Computations in Parallel Numerical Linear Algebra. *TR ICSI 91-060.*

[D88] Demmel, J., (1988). The Probability that a Numerical Analysis Problem is Difficult. *Mathematics of Computation, Vol.* **50**, *No. 182, pp 449-481.*

[EC91] Emiris, I., Canny, J., (1991). A General Approach to Removing Degeneracies. *Proc. 32nd Annual Symposium on Foundation of Computer Science, pp 405-413.*

[G86] von zur Gathen, J., (1986). Parallel Arithmetic Computations: A Survey. *Lecture Notes in Computer Science, Vol.* **233**, *Springer-Verlag, New-York, pp 93-122.*

[G82] Gray, A. (1982) Comparison Theorems for the Volumes of Tubes as Generalizations of the Weyl Tube Formula. *Topology, Vol* **21**, *No. 2, pp 201-228.*

[L52] La Vallee Poussin, C.J. de, (1952). Lecons sur l'Approximation des Fonctions d'une Variable reelle. *Paris: Gauthier-Villars.*

[MST89] Mansour, Y., Schieber, B., Tiwari, P., (1989). The Complexity of Approximating the Square Root. *Proc. 30th Annual Symposium on Foundation of Computer Science, pp 325-330.*

[N64] Newman, D.J., (1964). Rational Approximation to $|x|$. *Michigan Math. J.* **11**, *pp 11-14.*

[O85] Ocneanu, A., (1985). On the Volumes of Tubes about a Real Variety. *Report, Mathematical Sciences Research Institute, Berkeley.*

[PR85] V. Pan and J. Reif, (1985). Efficient Parallel Solution of Linear Systems. *Proc. 17th Annual ACM Symposium on Theory of Computing, pp 143-152.*

[R69] Rivlin, T.J., (1969). An Introduction to the Approximation of Function. *Dover Publications, New York, NY.*

[SS91] Shub, M., Smale, S., (1991). Complexity of Bezout's Theorem I. Geometric Aspects. *manuscript.*

[S90] Smale, S., (1990). Some Remarks on the Foundation of Numerical Analysis. *SIAM Review, Vol.* **32**, *No. 2, pp 211-220.*

[S90] Strassen, V., (1990). Algebraic Complexity Theory. *Handbook of Theoretical Computer Science, Vol. A, pages 635-672.*

[TWW88] Traub, J.F., Wasilkowski G.W., Wozniakowski H., (1988). Information Based Complexity. *Academic Press, New York.*

[V89] Vavasis, S., (1989). Gaussian Elimination with Pivoting is P-Complete. *SIAM J. Disc. Math., Vol.* **2**, *No.3, pp 413-423.*

[W39] Weyl, H., (1939). On the Volume of Tubes. *Amer. J. of Math. Vol* **61**, *pp 461-472.*

Optimal Filtering by Recurrent Neural Networks*

James Ting-Ho Lo
Department of Mathematics and Statistics
University of Maryland Baltimore County
Baltimore, MD 21228

Abstract

This paper demonstrates the methodology of synthesizing such a recurrent neural network into a filter by the network training. We call the resulting network a neural filter.

The training input/output data are the measurement/signal sequences, which are generated by computer simulation or actual experiment. No such assumptions as Markov properties, Gaussian distributions and additive/linear noises are required.

The neural filter is a recursive filter with the least error variance for the given network architecture. Moreover, it converges to the minimum variance filter in performance, as its size increases.

Several numerical examples are included, which show that the neural filters can equal the Kalman filter for a linear signal/sensor system and outperform the extended Kalman filters for nonlinear systems. In addition, the neural filters are shown to perform satisfactorily where the extended Kalman filter can not be reasonably derived.

I. Introduction

A generic problem in many applications, including aircraft/ship navigation, radar/sonar signal processing, image/vision analysis, missile guidance/tracking, robotics/control, etc., can be stated as follows: Let a random signal process x_t, $t = 1, 2, \cdots, T$ be causally observed by a measurement process y_t, $t = 1, 2, \cdots, T$, i.e., $y_t = h(x_1, \cdots, x_t, w_t)$, where h is a known or unknown Borel measurable function and w_t is a random noise sequence with known or unknown statistics. Find a fast algorithm to produce a good estimate $\widehat{\psi}(x_t)$ of a known function $\psi(x_t)$ of x_t given the measurements $y^t := \{y_s, 1 \leq s \leq t\}$.

The modern theory of optimal filtering[1, 3] is concerned mainly with such a problem with the signal and measurement processes described by

$$\begin{aligned} x_{t+1} &= f(x_t) + G(x_t)w_t, && \text{(I.1)} \\ y_t &= h(x_t) + v_t, && \text{(I.2)} \end{aligned}$$

where w_t and v_t are Gaussian noise processes with given statistics and f, G, and h are known measurable functions with such appropriate dimensions and properties that x and y exist and are unique.

In this paper, we will show how a multilayer perceptron (MLP) with output feedbacks can be synthesized into a good filter for the aforestated general problem. The synthesis

---

*This work was supported by Rome Laboratory under Contract No. F30602-91-C-0033.

is performed through the neural network training, in which y_t are used as the neural network inputs and x_t as the desired outputs. If a signal/sensor system model such as (I.1) and (I.2) is available, the training data y_t and x_t, are easily generated by computer simulation. Otherwise, the experimental data can be used instead. The training data are all that we need to synthesize the filter.

A requirement here that is not needed in the modern theory of optimal filtering is that x_t and y_t stay in a compact set. The fullfilment of it is easy to justify in the real world. The model, (I.1) and (I.2), violates this requirement. However, it is only an idealization, whose x_t and y_t may stray out of a compact region with arbitrarily small probability provided that the region is sufficiently large. Other than the requirement, no assumption such as the Markov property, the Gaussian distribution, or the linear dynamics is necessary.

The training process minimizes the mean square error between the network outputs $\widehat{\psi}(x_t)$ and the desired outputs $\psi(x_t)$. Consequently, after adequate training, the estimates produced by the network are minimum-variance for the given network architecture. Furthermore, it will be proven that when the size of network increases, the network converges to the minimum variance filter.

There are two types of feedback, namely teacher-forced feedbacks and free feedbacks. From the optimization point of view, teacher forcing amounts to constraints. From the statistical point of view, free feedbacks are allowed to record the most informative statistics at each time instant for the best estimation at the next time instant. Therefore, a free feedback is always better than a teacher-forced feedback, if the application condition allows its use. This point is borne out by our simulation results.

Four numerical examples are worked out and reported in this paper. Two of them show that the neural filters consistently outperform the extended Kalman filter and the iterated Kalman filter. The third shows that the performance of the neural filter is virtually indistinguishable from that of the Kalman filter for a linear system of (I.1) and (I.2). In the last example, neither the signal nor the measurement can be described by (I.1) or (I.2). Nevertheless, the neural filter works satisfactorily.

II. Multilayer Perceptrons as Neural Filters

The general feedback configuration considered in this paper has $j+k$ output nodes, among which $\alpha_1(t+1), \cdots, \alpha_k(t+1)$ are teacher-forced outputs and $\beta_1(t+1), \cdots, \beta_j(t+1)$ are free outputs at time $t+1$. The MLP is trained to have only the teacher-forced outputs mimic the desired outputs. All the free outputs and i of the teacher-forced outputs, say $\alpha_1(t+1), \cdots, \alpha_i(t+1)$, are delayed by one unit of time before being fed back to the input nodes, $\beta_t(t), \cdots, \beta_j(t), \alpha_1(t), \cdots, \alpha_i(t)$. In addition, the MLP has m input nodes on which the external inputs to the MLP, $\gamma_1(t+1), \cdots, \gamma_m(t+1)$, are clamped.

As discussed in the preceding section, by either computer simulation or experiments, we may have available a finite set S of signal/measurement sequences, $(x(t,\omega), y(t,\omega))$, $t = 1, \cdots, T$, $\omega \in S$. It is assumed that the set S is a random sample and adquately reflects the joint probability distributions of the signal and measurement prosesses $x(t)$ and $y(t)$.

Suppose that $\psi(x(t)) = [\psi_1(x(t)), \cdots, \psi_k(x(t))]^T$ are what we want to estimate and are thus the desired outputs for the actual outputs $\alpha_1(t), \cdots, \alpha_k(t)$. Using the m components of the measurement vector $y(t) = [y_1(t), \cdots, y_m(t)]^T$ as the external inputs, $\gamma_1(t), \cdots, \gamma_m(t)$, the training data consists of the I/O pairs $(y(t,\omega), \psi(x(t,\omega)))$, $t = 1, \cdots, T, \omega \in S$.

The training is to minimize, by the variation of the MLP weights w, the mean square error

$$C(w) = \frac{1}{T(\#S)} \sum_{\omega \in S} \sum_{t=1}^{T} \sum_{l=1}^{k} (\alpha_l(t,\omega) - \psi_l(x(t,\omega)))^2 \tag{II.1}$$

where $\#S$ is the number of elements in S and $\alpha_l(t,\omega)$ is the actual MLP output corresponding to the input sequence $y(\tau,\omega)$, $\tau = 1, \cdots, t$, at time t. We note here that $\alpha_l(t,\omega)$ is a function of $y(\tau,\omega)$, $\tau = 1, \cdots, t$.

A good training method, which is used in our simulation, is first to evaluate the gradient of the above error function with respect to w by the time-dependent recurrent backpropagation by Werbos[5] and Pearlmutter[4] and then to use it in a conjugate gradient algorithm for optimization. An adequately trained network through minimizing $C(w)$ will be called a neural filter.

III. Convergence to the Minimum Variance Filter

The collection of all k-dimensional random vectors $x = [x_1, \cdots, x_k]^T$ is a Hilbert space L_2 with the norm $E[\| x \|^2] = E[\sum_{l=1}^{k} x_l^2]$, where E denotes the expectation or the average over the sample space Ω with the σ-field $\mathcal{A}$ of events and the probability measure P.

Under the assumption that averaging over S is indistinguishable from taking expectation over Ω, we see that

$$\begin{aligned} C(w) &= \frac{1}{T(\#S)} \sum_{\omega \in S} \sum_{t=1}^{T} \| \psi(x(t,\omega)) - E[\psi(x(t))|y^t(\omega)] \|^2 \\ &+ \frac{1}{T(\#S)} \sum_{\omega \in S} \sum_{t=1}^{T} \| \alpha(t,\omega) - E[\psi(x(t))|y^t(\omega)] \|^2 \end{aligned}$$

Training amounts to minimizing the second term above. The objective is, of course, to minimize $(1/T) \sum_{t=1}^{T} E[\| \alpha(t) - \psi(x(t)) \|^2]$.

Recall the neural filter architecture described in Sec. 2. We observe that adding a neuron to a hidden layer decreases $\min_w C(w)$ and so does adding a feedback, free or teacher-forced. We are now ready to state and prove the main theorem of this paper.

Theorem. Consider the n-dimensional random process $x(t)$ and the m-dimensional random process $y(t)$, $t = 1, \cdots, T$ defined on a probability space $(\Omega, \mathcal{A}, P)$. Assume that the range $\Lambda := \{y(t,\omega) | t = 1, \cdots, T,\ \omega \in \Omega\} \subset R^m$ is compact and ψ is an arbitrary k-dimensional Borel function of $x(t)$ such that the second moments $E[\| \psi(x(t)) \|^2]$, $t = 1, \cdots, T$, are finite. Let $\alpha(t)$ denote the k-dimensional output at time t of the neural filter which has taken the external inputs, $y(1), \cdots, y(t)$, in the given order.

1. Given $\epsilon > 0$, there exist a neural filter of the architecture described in Sec. 1 such that

$$\frac{1}{T} \sum_{t=1}^{T} E[\| \alpha(t) - E[\psi(x(t))|y^t] \|^2] < \epsilon.$$

2. If the MLP in the neural filter described in Sec. 2 has a single hidden layer with N neurons, $i = 0$ teacher-forced feedback nodes, and $j = m(T-1)$ free feedback nodes and $\alpha(t)$ is written as $\alpha(t; N)$ here to indicat its dependency on N, then

$$r(N) := \min_{w} \frac{1}{T} \sum_{t=1}^{T} E[\| \alpha(t, N) - E[\psi(x(t))|y^t] \|^2] \tag{III.1}$$

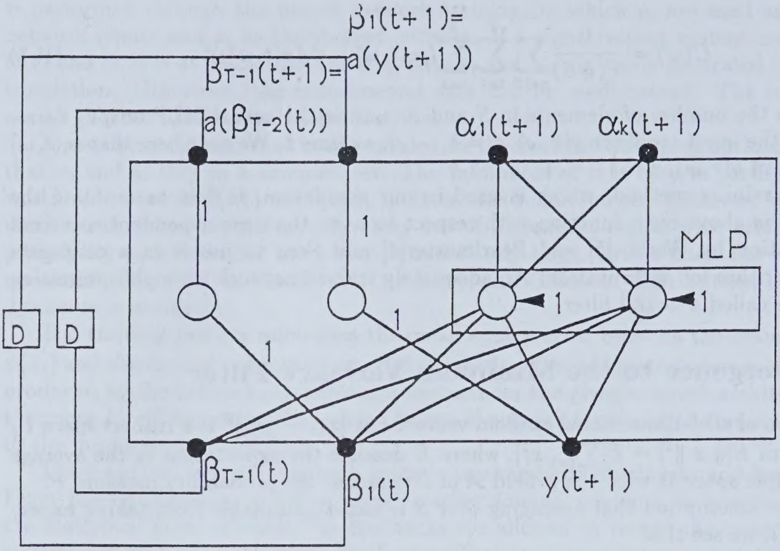


Figure 1: A special feedback configuration used in the proof of the main theorem

is monotone decreasing and converges to 0 as N approaches infinity.

Proof. Since (1) is an immediate consequence of (2), we shall only prove (2).

It is obvious that $r(N)$, $N = 1, 2, \cdots$, are all bounded from below by 0, the sequence $r(N)$ converges and we denote the limit by $r(\infty)$. To prove that $r(\infty) = 0$, it suffices to show that for any $\epsilon > 0$, there is an integer M such that $|r(M)| < \epsilon$. This will be shown in the following for the case in which the measurement process $y(t)$ is scalar-valued, i.e. $m = 1$. The proof for $m > 1$ is similar and omitted.

As illustrated in Fig. 1, the $T - 1$ free feedback loops can be so constructed that for $\tau = 1, \cdots, T - 1$,

$$\beta_\tau(t+1) = a(\beta_{\tau-1}(t)),\ t = 0, , \cdots, T-1. \tag{III.2}$$

The free feedback output nodes are initialized by $\beta_\tau(0) = 0$, $\tau = 1, \cdots, T-1$. At time $t + 1$, the free feedback input nodes are

$$\begin{aligned}&(\beta_{T-1}(t), \cdots, \beta_{t+1}(t), \beta_t(t), \beta_{t-1}(t), \cdots, \beta_1(t))\\ =\ &(0, \cdots, 0, a^{\circ t}(y(1)), a^{\circ(t-1)}(y(2)), \cdots, a(y(t))),\end{aligned}$$

where $a^{\circ t}$ denote the t-fold composition $a \circ a \circ \cdots \circ a$ of a. Including both the feedback input nodes and the external input node, we have the T-dimensional MLP input vector at time $t + 1$,

$$z(t+1) = [\beta_{T-1}(t), \cdots, \beta_1(t), y(t+1)]^T.$$

Recall that

$$\zeta(t+1) := E[\psi(x(t+1))|a^{\circ t}(y(1)), a^{\circ(t-1)}(y(2)), \cdots, a(y(t)), y(t+1)]$$

is a Borel measurable function of $a^{\circ t}(y(1)), \cdots, a(y(t))$, and $y(t+1)$. Let us denote it by $\zeta(t+1) = \lambda_{t+1}(a^{\circ t}(y(1)), \cdots, a(y(t)), y(t+1))$. Now let us consider the mapping

$f : R^T \to R^k : \xi \mapsto \eta = f(\xi)$ defined by

$$\begin{aligned}\eta &= f(\xi_1,\cdots,\xi_T)\\ &= \begin{cases}\lambda_1(\xi_T), & \text{if } \xi_1=\cdots=\xi_{T-1}=0, \xi_T\neq 0 \text{ and } \lambda_1(\xi_T) \text{ is defined;}\\ \lambda_2(\xi_{T-1},\xi_T), & \text{if } \xi_1=\cdots=\xi_{T-2}=0,\xi_{T-1}\neq 0,\xi_T\neq 0\\ & \text{and } \lambda_2(\xi_{T-1},\xi_T)\text{ is defined;}\\ \vdots & \\ \lambda_T(\xi_1,\cdots,\xi_T), & \text{if } \xi_1\neq 0,\cdots,\xi_T\neq 0 \text{ and } \lambda_T(\xi_1,\cdots,\xi_T) \text{ is defined;}\\ 0, & \text{otherwise.}\end{cases}\end{aligned}$$

Let us consider also the measures μ_t defined on the Borel sets in R^T by

$$\begin{aligned}\mu_1(d\xi_1\times d\xi_2\times\cdots\times d\xi_T) &= \begin{cases}\frac{1}{T}P(\omega|y(1)\in d\xi_T), & \text{if } 0\in d\xi_t, \text{ for } t=1,\cdots,T-1,\\ 0, & \text{otherwise;}\end{cases}\\ \mu_2(d\xi_1\times d\xi_2\times\cdots\times d\xi_T) &= \begin{cases}\frac{1}{T}P(\omega|a(y(1))\in d\xi_{T-1}, y(2)\in d\xi_T), & \text{if } 0\in d\xi_t, \text{ for}\\ & t=1,\cdots,T-2,\\ 0, & \text{otherwise;}\end{cases}\\ &\ \ \vdots\\ \mu_T(d\xi_1\times d\xi_2\times\cdots\times d\xi_T) &= \frac{1}{T}P(\omega|a^{\circ(T-1)}(y(1))\in d\xi_1,\cdots,a(y(T-1))\in d\xi_{T-1}),\\ &\quad y(T)\in d\xi_T).\end{aligned}$$

Notice that $\mu=\mu_1+\cdots+\mu_T$ is also a mesure on the Borel sets in R^T and that

$$\begin{aligned}&\int\|f(\xi_1,\cdots,\xi_T)\|^2\mu(d\xi_1\times d\xi_2\times\cdots\times d\xi_T)\\ &\quad= \frac{1}{T}\{\int\|\lambda_1(\xi_T)\|^2\mu_1(d\xi_1\times d\xi_2\times\cdots\times d\xi_T)\\ &\qquad+\int\|\lambda_2(\xi_{T-1},\xi_T)\|^2\mu_2(d\xi_1\times d\xi_2\times\cdots\times d\xi_T)\\ &\qquad+\cdots\\ &\qquad+\int\|\lambda_T(\xi_1,\cdots,\xi_T)\|^2\mu_T(d\xi_1\times d\xi_2\times\cdots\times d\xi_T)\}\\ &\quad= \frac{1}{T}\{E[\|E[\psi(x(1))|y(1)]\|^2]+E[\|E[\psi(x(2))|a(y(1)),y(2)]\|^2]\\ &\qquad+\cdots+E[\|E[\psi(x(T))|a^{\circ(T-1)}(y(1)),a^{\circ(T-2)}(y(2)),\cdots,y(T)]\|^2]\}\\ &\quad\le \frac{1}{T}\{E[E[\|\psi(x(1))\|^2|y(1)]]+E[E[\|\psi(x(2))\|^2|a(y(1)),y(2)]]\\ &\qquad+\cdots+E[E[\|\psi(x(T))\|^2|a^{\circ(T-1)}(y(1)),a^{\circ(T-2)}(y(2)),\cdots,y(T)]]\}\\ &\quad= \frac{1}{T}\{E[\|\psi(x(1))\|^2]+E[\|\psi(x(2))\|^2]+\cdots+E[\|\psi(x(T))\|^2]\}\\ &\quad< \infty.\end{aligned}$$

Since $y(t)$, $t=1,\cdots,T$, are assumed to have compact ranges, it is easy to see that $a^{\circ t}(y(\tau))$, t and $\tau=1,\cdots,T$, all have compact ranges. Hence there is a compact set W in R^T such that $\mu(W)=1$.

Consider now the normed space $L_2(R^T,\mu)$ with the norm defined by $[\int\|g(\xi_1,\cdots,\xi_T)\|^2\ \mu(d\xi_1\times\cdots\times d\xi_T)]^{1/2}$ for each k-dimensional function $g\in L_2(R^T,\mu)$. By Corollary 2.2 of the paper [2] by Hornik, Stinchcombe and White, the k-dimensional functions represented by all the MLPs with k output nodes and a single layer of hidden neurons are dense in $L_2(R^T, \mu)$. It then follows that for any $\epsilon > 0$, there is such an MLP that the function $g(\xi_1, \cdots, \xi_T)$ that it represents satisfies

$$\int \|f(\xi_1, \cdots, \xi_T) - g(\xi_1, \cdots, \xi_T)\|^2 \mu(d\xi_1 \times \cdots \times d\xi_T) < \epsilon.$$

Let us translate the left hand side of this inequality from $L_2(R^T, \mu)$ back to the probability space $(\Omega, \mathcal{A}, P)$:

$$\begin{aligned}
&\int \|f(\xi_1, \cdots, \xi_T) - g(\xi_1, \cdots, \xi_T)\|^2 \mu(d\xi_1 \times d\xi_2 \times \cdots \times d\xi_T) \\
&= \frac{1}{T}\{E[\|E[\psi(x(1))|y(1)] - g(0, \cdots, 0, y(1))\|^2] \\
&\quad + E[\|E[\psi(x(2))|a(y(1)), y(2)] - g(0, \cdots, 0, a(y(1)), y(2))\|^2] \\
&\quad + \cdots \\
&\quad + E[\|E[\psi(x(T))|a^{\circ(T-1)}(y(1)), a^{\circ(T-2)}(y(2)), \cdots, y(T)] - g(a^{\circ(T-1)}(y(1)), \\
&\quad a^{\circ(T-2)}(y(2)), \cdots, y(T))\|^2]\} \\
&= \frac{1}{T}\sum_{t=1}^{T} E[\|E[\psi(x(t))|y^t] - g(0, \cdots, 0, a^{\circ(t-1)}(y(1)), a^{\circ(t-2)}(y(2)), \cdots, y(t))\|^2]
\end{aligned}$$

where the last equality holds, because the σ-fields generated by y^t and $\{a^{\circ(t-1)}(y(1)), a^{\circ(t-2)}(y(2)), \cdots, y(t)\}$ are identical due to the monotonicity of a.

We notice that the foregoing approximation network $g(\xi_1, \cdots, \xi_T)$ and the feedback network (III.2) discussed earlier form an MLP with a single hidden layer. The number N of hidden neurons of this MLP is the sum of those of $g(\xi_1, \cdots, \xi_T)$ and of (III.2). Recalling (III.1) we have

$$\begin{aligned}
r(N) \leq & \frac{1}{T}\sum_{t=1}^{T} E[\|E[\psi(x(t))|y^t] - g(0, \cdots, 0, a^{\circ(t-1)}(y(1)), \\
& a^{\circ(t-2)}(y(2)), \cdots, y(t))\|^2] < \epsilon,
\end{aligned}$$

since $r(N)$ minimizes the error function in (III.1) by the variation of the weights of the MLP of exactly the same architectures as that for g. This completes the proof. □

IV. Numerical Examples

The noise processes, w and v used in the sequal are independent standard white Gaussian sequences and the neural filter has 0 teacher-forced feedback, 1 free feedback, and a single hidden layer of 7 neurons.

Example 1. The signal/sensor system is

$$\begin{aligned}
x_{t+1} &= 1.1\exp(-2x_t^2) - 1 + 0.5w_t, \\
y_t &= x_t^3 + 0.1v_t,
\end{aligned}$$

where x_0 is Gaussian with mean -0.5 and variance 0.1^2. Note that $x_{t+1} = 1.1\exp(-2x_t^2) - 1$ has a global attractor at $x_t = 0.0844$.

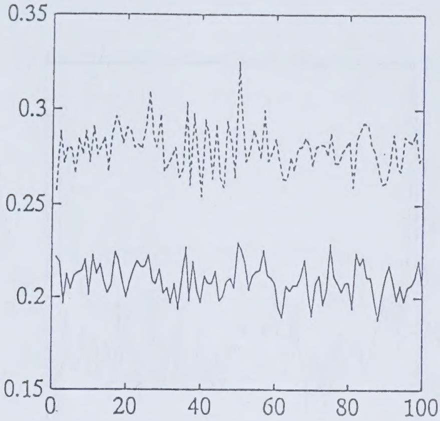


Figure 2: RMSE of the IEKF (---) and the neural filter (—) versus time for Example 1.

The EKF fails badly. The RMSEs for the IEKF (---) and the neural filter (—) for 500 test runs are shown in Fig. 2.

Example 2. The signal/sensor system is

$$\begin{aligned} x_{t+1} &= 1.7\exp(-2x_t^2) - 1 + 0.1w_t, \\ y_t &= x_t^3 + 0.1v_t, \end{aligned}$$

where x_0 is Gaussian with mean zero and variance 0.5^2. Note that $x_{t+1} = 1.7\exp(-2x_t^2) - 1$ is a chaotic process.

The RMSEs for the EKF $(\cdots)$, the IEKF (---) and the neural filter (—) for 500 test runs are shown in Fig. 3.

Example 3. The signal/sensor system is

$$\begin{aligned} x_{t+1} &= 0.9x_t + 0.2w_t, \\ y_t &= x_t + v_t, \end{aligned}$$

where x_0 is Gaussian with mean zero and variance 1^2.

The RMSEs for the Kalman filter (---) and the neural filter (—) for 500 test runs are shown in Fig. 4.

Example 4. The signal/sensor system is

$$\begin{aligned} x_{t+1} &= 0.5x_t + 0.5\tanh(x_t + 0.5w_t), \\ y_t &= x_t + 0.5x_t^3 v_t, \end{aligned}$$

where x_0 is Gaussian with mean zero and variance 0.5^2. Note that neither the signal nor the measurement process can be described by a conventional signal/sensor system model.

The RMSEs for the neural filter for 500 test runs are shown in Fig. 5. The true signal (—) and its estimate (---) produced by the neural filter for a single run are shown in Fig. 6.

Acknowledgement

The author would like to acknowledge with thanks Dr. Lei Yu from the Maryland Technology Corporation for training and simulating the neural networks, and Mr. Daniel

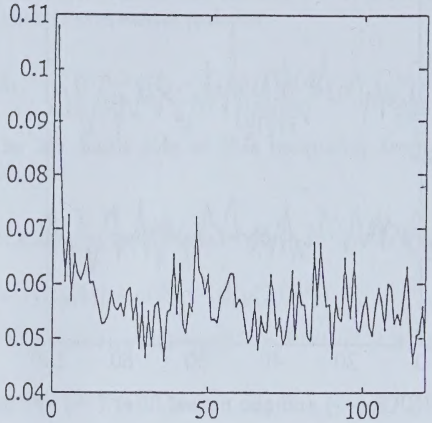


Figure 5: RMSE of the neural filter versus time for Example 4.

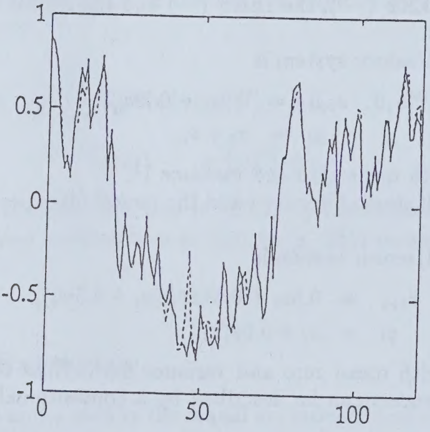


Figure 6: The true signal (—) and its estimate (---) produced by the neural filter versus time for Example 4.

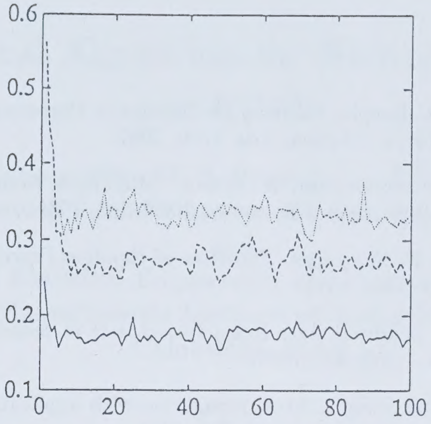


Figure 3: RMSE of the EKF (···), the IEKF (---) and neural filter (—) versus time for Example 2.

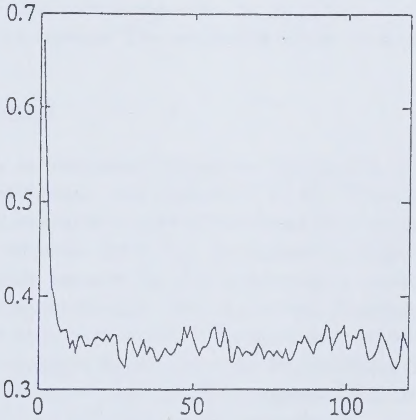


Figure 4: RMSE of the Kalman filter (---) and the neural filter (—) versus time for Example 3.

J. Gentile and Dr. Vincent Vanicola from the Rome Laboratory for their support in many forms.

REFERENCES

[1] R. S. Bucy and P. D. Joseph. *Filtering for Stochastic Processes with Applications to Guidanc, Second Edition.* Chelsea, New York, 1987.

[2] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[3] R. S. Liptser and A. N. Shiryayev. *Statistics of Random Processes I: general theory.* Springer-Verlag, New York, 1977.

[4] B. A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269, 1989.

[5] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1:339–356, 1988.

A Hierarchical Algorithm for Neural Training and Control

T. V. Theodosopoulos, M. S. Branicky, and M. M. Livstone
Laboratory for Information and Decision Systems
and
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA

Abstract

Lately, there has been an extensive interest in the possible uses of neural networks for nonlinear system identification and control. In this paper, we provide a framework for the simultaneous identification and control of a class of unknown, uncertain nonlinear systems. The identification portion relies on modeling the system by a neural network which is trained via a local variant of the Extended Kalman Filter. We will discuss this local algorithm for training a neural network to approximate a nonlinear feedback system. We also give a dynamic programming-based method of deriving near optimal control inputs for the real plant based on this approximation and a measure of its error (covariance). Finally, we combine these methods in a hierarchical algorithm for identification and control of a class of uncertain, unknown systems. The complexity of the whole algorithm is analyzed.

1 Introduction

Lately, there has been an extensive interest in the possible uses of neural networks for nonlinear system identification and control [7, 3, 6]. Typically, these approaches use multilayer feedforward neural networks with various backpropagation learning rules.

Recurrent neural networks have thus far enjoyed a disproportionately low level of attention. That is mainly because the delays internal to such a model make traditional backpropagation learning ineffective. Matthews and Moschytz [5] suggest that just as multilayer perceptrons have been shown to be good approximators for nonlinear feedforward systems, so can recurrent neural networks for nonlinear feedback systems.

In particular, Matthews and Moschytz first suggested the use of the Extended Kalman Filter (EKF) algorithm for training neural networks [5]. Later Livstone, Farrell, and Baker showed how to localize the EKF algorithm to make the learning more computationally efficient [4]. The Spatially Localized Extended Kalman (SLEK) algorithm of [4] forms the basis of this paper.

We will first discuss the SLEK algorithm for training a neural network to approximate a nonlinear feedback system. We next give a dynamic programming-based method of deriving near optimal control inputs for the real plant based on this approximation and a measure of its error (covariance). Combining these methods leads to a hierarchical algorithm for identification and control of a class of uncertain, unknown systems. Finally, the complexity of the whole algorithm is analyzed.

2 Identification and Control of Unknown, Uncertain Systems

2.1 Approximating the System

The algorithms in this paper are written to handle the case where

$$\begin{aligned} x[k+1] &= f(x[k],u[k]) + f_{\text{known}}(x[k],u[k]) + g(\xi[k]) \\ y[k] &= h(x[k]) + v[k] \end{aligned}$$

where $x, \xi \in R^n$, $u \in R^m$, $y, v \in R^p$ and f, f_{known}, g, h are continuous, possibly nonlinear, functions on the associated spaces. We assume f_{known}, g, and h are **known**; f is **unknown**. Further, $\xi[k]$ and $v[k]$ are uncorrelated white Gaussian noise sequences that are also uncorrelated with the initial condition $x[0]$.

However, for ease of presentation, we will treat the special case where the system dynamics are modeled as:

$$\begin{aligned} x[k+1] &= f(x[k],u[k]) + \xi[k] \\ y[k] &= x[k] + v[k] \end{aligned}$$

where $x, \xi, v, y \in R^n$, $u \in R^m$, and $f \in C[R^{n+m}, R^n]$; f is **unknown**. Further, $\xi[k]$ and $v[k]$ are uncorrelated white Gaussian noise sequences with covariances

$$\begin{aligned} E\{\xi[k]\xi^T[n]\} &= Q[k]\delta_{kn} \\ E\{v[k]v^T[n]\} &= R[k]\delta_{kn} \\ E\{\xi[k]v^T[n]\} &= 0 \end{aligned}$$

and $\xi[k], v[k]$ are uncorrelated with the initial state $x[0]$. Note that the measurement equation is linear. The nonlinear state equation will be approximated by the network

$$x[k+1] \approx f_{\text{net}}(x[k], u[k]; \theta[k])$$

where $\theta[k]$ are network parameters to be learned. Let $q[k] = [x^T[k], u^T[k]]^T$. We will refer to this R^{n+m} vector space as the *state-input space*. With this notation we see

$$f_{\text{net},i}(q[k];\theta[k]) = \sum_{j=1}^{N} A_{ij} g_j(q[k]) \tag{1}$$

where the g_j are radial basis functions (RBFs) and A_{ij}, the magnitudes or *weights*, are the parameters in $\theta[k]$. In this paper, we will use Gaussian RBFs to simplify calculations:

$$g_j(q[k]) = \exp\left\{-(q[k]-c_j)^T S^{-1}(q[k]-c_j)\right\}$$

where the covariance matrix $S = \text{diag}(s_1, s_2, \ldots, s_{n+m})$ and c_j is the jth mean or *center*. Finally, we have the parameter vector

$$\theta = [A_{11}, A_{21}, \ldots, A_{n1}, A_{12}, A_{22}, \ldots, A_{n2}, \ldots, A_{1N}, A_{2N}, \ldots, A_{nN}]^T$$

Thus, θ is a vector in R^{nN}.

Implicitly, we have assumed that the state-space is really $\mathcal{S} \subset R^n$ and the input space is $\mathcal{C} \subset R^m$, where $\mathcal{S}$ and $\mathcal{C}$ are compact sets surrounding the region of interest in state and input space, respectively. For many applications—*e.g.*, motor control or robotics—the state space is inherently compact (*e.g.*, angles, velocity saturation, kinematic constraints). In practice, one would use a safety net controller (usually based on that portion of dynamics that is known *a priori*) to insure that the state remains within $\mathcal{S}$.

2.2 SLEK Algorithm

One could now use the EKF algorithm to simultaneuously approximate the state and learn the parameters (here, weights) [5]. However, this requires $O(n^3 + (nN)^3)$ operations per step (assuming no cross-correlation between the state and the parameters).

Instead, we wish to use a localized version of the EKF to update at each step only those parameters which are significant to the model at that point. When the underlying basis functions have compact support (or die quickly) as in the case of Guassian RBFs, this strategy can lead to an algorithm which is more computationally tractable. The so-called Spatially Localized Extended Kalman Filter (SLEK) algorithm of [4] implements such a strategy.

To localize, we need to create a generalized nearest neighbor map:

$$\begin{aligned} T_\epsilon : R^{n+m} &\longrightarrow \text{power set of } \{1, 2, \ldots, nN\} \\ T_\epsilon(q) &= \left\{ j : \left| \left. \frac{\partial f_{\text{net},i}}{\partial \theta_j} \right|_q \right| > \epsilon \right\} \end{aligned}$$

That is, $T_\epsilon(q)$ is a map from a point in the state-input space to the indices of those parameters that are significant to the approximation at that point. Next, we will let $\phi_{\text{local}}(q)$ denote the vector (matrix) obtained by ordering those elements of the vector (matrix) ϕ which are "picked off" by $T_\epsilon(q)$. For example, $\theta_{\text{local}}(q) \in R^p$ where p is the cardinality of $T_\epsilon(q)$ and $\theta_{\text{local},j}(q) = \theta_k$ if k is the jth smallest element of $T_\epsilon(q)$. Likewise, the matrix $\Theta_{\text{local}}(q)$ would be $p \times p$ with Θ_{ij} part of $\Theta_{\text{local}}(q)$ if $\{i, j\} \subset T_\epsilon(q)$ (with the obvious ordering). Finally, in a slight abuse of notation, we will say $\theta_i \in \theta_{\text{local}}(q)$ whenever $i \in T_\epsilon(q)$. Specifically, we will say $A_{ij} \in \theta_{\text{local}}(q)$ whenever $ij \in T_\epsilon(q)$.

To ease exposition and the complexity analysis, we will consider the nearest neighbor map that gives the two closest centers in each dimension. In other words, if d_i is the spacing of the centers of the RBFs in dimension i of the state-input space and if c_j is the center vector of the jth RBF,

$$\theta_{\text{local}}(q) = \{A_{kj} : |q_i - c_{j,i}| < d_i\}$$

where $k = 1, 2, \ldots, n$, $j = 1, 2, \ldots, N$, and $i = 1, 2, \ldots, n + m$. Note that this can be made consistent with our general $(T_\epsilon(q))$ notation by picking

$$\epsilon \leq \exp\left\{ -\sum_{i=1}^{n+m} (d_i / s_i)^2 \right\}$$

So now the sum in (1) is taken over j such that $A_{ij} \in \theta_{\text{local}}(q[k])$. Next, let's construct an augmented state vector with the local parameters of θ:

$$z[k+1] = \begin{bmatrix} x[k+1] \\ \theta_{\text{local}}[k+1](q[k]) \end{bmatrix}$$

$$
\begin{aligned}
&\approx \begin{bmatrix} f_{\text{net}}(x[k],u[k]) \\ \theta_{\text{local}}[k](q[k]) \end{bmatrix} + \begin{bmatrix} \xi[k] \\ \eta_{\text{local}}[k] \end{bmatrix} \\
&= F(z[k],u[k]) + w[k]
\end{aligned}
$$

where

$$
\begin{aligned}
E\{\eta[k]\eta^T[n]\} &= \Theta[k]\delta_{kn} \\
E\{\eta[k]\xi^T[n]\} &= 0
\end{aligned}
$$

so that

$$
E\{w[k]w^T[n]\} = \delta_{kn} \begin{bmatrix} Q[k] & 0 \\ 0 & \Theta_{\text{local}}[k] \end{bmatrix}
$$

If we linearize the new system about the current conditions we see that

$$
\delta z[k+1] \approx A[k]\delta z[k] + B[k]\delta u[k] + w[k]
$$

where

$$
\begin{aligned}
A[k] &= \left.\frac{\partial F}{\partial z}\right|_{\hat{z}[k|k],u[k]} = \begin{bmatrix} \partial f_{\text{net}}/\partial x & \partial f_{\text{net}}/\partial \theta_{\text{local}} \\ 0 & I \end{bmatrix}_{\hat{z}[k|k],u[k]} \\
B[k] &= \left.\frac{\partial F}{\partial u}\right|_{\hat{z}[k|k],u[k]} = [\partial f_{\text{net}}/\partial u]_{\hat{z}[k|k],u[k]}
\end{aligned}
$$

where $\hat{z}[k|k]$ is our best current estimate of the state provided by the SLEK algorithm (see below). With some straightforward calculations, we see that

$$
\left.\frac{\partial f_{\text{net},i}}{\partial q_l[k]}\right|_{q[k]} = \frac{2}{s_l^2}\left[h_{\text{net},(i,l)}(q[k]) - q_l[k] f_{\text{net},i}(q[k])\right]
$$

where

$$
h_{\text{net},(i,l)}(q[k]) = \sum_{j:A_{ij}\in\theta_{\text{local}}(q[k])} A_{ij} c_{j,l} g_j(q[k])
$$

is a network with the same RBFs but with scaled weights. We also have:

$$
\left.\frac{\partial f_{\text{net},i}}{\partial \theta_j[k]}\right|_{q[k]} = \begin{cases} 0, & \theta_j[k] \notin \theta_{\text{local}}(q[k]) \\ g_l(q[k]), & \theta_j[k] \in \theta_{\text{local}}(q[k]), l = \lceil j/n \rceil \end{cases}
$$

The measurement equation is $y[k] = [I,\ 0]z[k] + v[k]$.

Now, we will use the SLEK algorithm to estimate the augmented state vector. The algorithm is as follows (see [4, 1]):

$$
\begin{aligned}
\hat{z}[k|k-1] &= F(\hat{z}[k-1|k-1], u[k-1]) \\
\Sigma[k|k-1] &= A[k-1]\Sigma[k-1|k-1]A^T[k-1] + \begin{bmatrix} Q[k] & 0 \\ 0 & \Theta_{\text{local}}[k] \end{bmatrix} \\
H[k] &= \Sigma[k|k-1]C^T\left[C\Sigma[k|k-1]C^T + R[k]\right]^{-1} \\
\hat{z}[k|k] &= \hat{z}[k|k-1] + H[k]\left[y[k] - CF(\hat{z}[k|k-1], u[k])\right] \\
\Sigma[k|k] &= [I - H[k]C]\,\Sigma[k|k-1]
\end{aligned}
$$

Note that the calculations above can be performed much more efficiently than the general $n+2^n$ Kalman Filter algorithm because of the special form of C and $\Sigma[\cdot|\cdot]$: $C = [I_n,\ 0_{n,2^n}]$

and $\Sigma[\cdot|\cdot] = \text{blockdiag}(\Sigma_1[\cdot|\cdot], \Sigma_2[\cdot|\cdot])$, where $\Sigma_i[\cdot|\cdot]$ are symmetric and have dimensions $n \times n$ and $2^n \times 2^n$, respectively.

The successive updates of the SLEK algorithm attempt to estimate the augmented state vector, part of which are the local parameters of the neural network. Thus the SLEK algorithm trains the neural network by finding the weights necessary to express the dynamic nonlinearity as a local sum of Gaussians (assuming such a decomposition is possible).

2.3 Near Optimal Control with an Approximate Model

Now we proceed with the control problem. At every discrete point in time we have an estimate of where the system will go if we were to apply a specific input. This estimate is provided to us by the SLEK algorithm along with a measure of its uncertainty, the covariance ($\Sigma_1[k|k]$). Let

$$G : R^{n+m} \to R$$

be a (non-negative) cost function. For a discount factor $\alpha \in (0,1)$ we define

$$J(x[0]) = E\left\{\sum_{i=0}^{\infty} \alpha^i G(x[i], u[i])\right\}$$

to be the penalty function for the infinite horizon problem. So we want to

$$\min_{u[0],u[1],u[2],\ldots} J(x[0])$$

with

$$\Pr(x[k+1] = y|q[k]) = \begin{cases} K(n,\delta)\ |\Sigma_1[k|k]|^{-1/2}\ \Gamma_k(y), & \|f_{\text{net}}(q[k];\theta[k]) - y\|_\infty \leq \delta \\ 0, & \text{otherwise} \end{cases}$$

where $|A|$ is the determinant of matrix A and $K(n,\delta)$ is a constant that normalizes the truncated Gaussian pdf to integrate to unity.

$$\Gamma_k(y) = \exp\left\{-\frac{1}{2}\left[y - f_{\text{net}}(q[k];\theta[k])\right]^T (\Sigma_1[k|k])^{-1} \left[y - f_{\text{net}}(q[k];\theta[k])\right]\right\}$$

and δ comes from discretizing the state space. In particular, if $\mathcal{S} \subset R^n$ was the state space before, we now consider the δ-lattice that spans $\mathcal{S}$, called $\mathcal{S}^\delta$. Thus we substitute $\mathcal{S}$ with a grid of "fineness" δ. From the definition of the probability, we only allow nonzero probability of the two closest neighbors in each dimension of the state estimate (though, as with T_ϵ before, more general maps can be formulated). This is a convention for computational tractability.

From now on, the input space is also assumed to be discrete (for practical considerations). In particular the input space is $\mathcal{C} \subset R^m$. The discretization, $\mathcal{C}^{\delta'}$, need not have $\delta' = \delta$ because it arises from different considerations.

We will use dynamic programming to solve the above control problem. In particular, we will choose:

$$u[k+1] = \arg\min_{u \in \mathcal{C}} \left\{ G(x[k], u) + \alpha \sum_{y \in \mathcal{S}^\delta} \Pr(y|x[k], u)\hat{J}(y) \right\}$$

where $\hat{J}(y)$ is the estimated *cost-to-go.* So $u[k+1]$ is the solution of

$$\left.\frac{\partial G}{\partial u}\right|_{u[k+1]} = -\alpha \sum_{y \in \mathcal{S}^\delta} \left.\frac{\partial \Pr(y|q[k])}{\partial u}\right|_{u[k+1]} \hat{J}(y) \tag{2}$$

We now digress to discuss how to estimate the cost-to-go. Let $B(\mathcal{S}^\delta)$ be the set of bounded continuous functions on $\mathcal{S}^\delta$. Consider the following operator:

$$\begin{aligned} T : B(\mathcal{S}^\delta) &\rightarrow B(\mathcal{S}^\delta) \\ TJ(x) &= \min_{u \in \mathcal{C}} \left\{ G(x,u) + \alpha \sum_{y \in \mathcal{S}^\delta} \Pr(y|q[k]) J(y) \right\} \end{aligned}$$

This operator can be proven to be a contraction [2]. So, since $B(\mathcal{S}^\delta)$ is complete, T has a unique fixed point, which is the solution we seek, *i.e.*, the real cost-to-go. So, if we start from some approximation $J_0(x)$, we can iterate:

$$J_{i+1}(x) = TJ_i(x)$$

One could even use fancier techniques to obtain tight bounds on the error at every iteration (*e.g.*, see [8]). We will not concern ourselves with the specifics of this approximation in this paper.

Going back to solving (2), we see that it is not a straightforward task. So we will iterate $\hat{u}$ to find $u[k+1]$. We will first introduce some notation:

$$\begin{aligned} c'_r &= [c_{r,n+1}, \ldots, c_{r,n+m}]^T \\ d' &= [d_{n+1}, \ldots, d_{n+m}]^T \\ B_r &= \left\{ y \in R^m \mid \|y - c'_r\|_\infty \leq \frac{1}{2} d' \right\} \end{aligned}$$

Also, define the function

$$\begin{aligned} \rho_r : R^m &\rightarrow B_r \\ \rho_r(x) &= \arg\min_{y \in B_r} \|x - y\|_\infty \end{aligned}$$

Then

$$\hat{u}_r^{(p+1)}(x) = \rho_r \left(\hat{u}_r^{(p)}(x) + \beta \left[\left.\frac{\partial G}{\partial u}\right|_{\hat{u}_r^{(p)}(x)} + \alpha \sum_{y \in \mathcal{S}^\delta} \left.\frac{\partial \Pr(y|x,u)}{\partial u}\right|_{\hat{u}_r^{(p)}(x)} J_k(y) \right] \right)$$

with $\beta < 0$, where

$$\begin{aligned} \left.\frac{\partial \Pr(y|x,u)}{\partial u}\right|_{\hat{u}_r^{(p)}(x)} &= \Pr(y|x, \hat{u}_r^{(p)}(x)) \Delta_k(y) \\ \Delta_k(y) &= [y - f_{\text{net}}(x, \hat{u}_r^{(p)}(x); \theta[k])]^T (\Sigma_1[k|k])^{-1} \left.\frac{\partial f_{\text{net}}}{\partial u}\right|_{\hat{u}_r^{(p)}(x)} \end{aligned}$$

Above, p is the index of iteration and r shows that this gradient search is implemented in parallel N^m times, where N is the number of RBFs per input dimension. In particular, for each r we will start the corresponding search from c'_r and limit it to the $\|\cdot\|_\infty$ ball around it of radius d'.

2.4 The Hierarchical Algorithm

We now describe the overall algorithm (see Figure 1):

0. Initialize the algorithm with $k = 0$ and $u[-1] = 0$; $\hat{u}_r^{(0)}(x)$ of each D. P. r at c'_r; and the cost-to-go estimate at some $\hat{J}_0(x)$, for all $x \in \mathcal{S}^\delta$
1. Given $y[k-1], u[k-1]$, run SLEK at the top level, updating f_{net}, h_{net}, and $\Sigma_1[k-1|k-1]$ and send them to the lower level.
2. Given f_{net}, h_{net}, and $\Sigma_1[k-1|k-1]$, run at the D. P. command level for each $x \in \mathcal{S}^\delta$
$$\hat{J}_{k-1}(x) = G(x, u_{\min}(x)) + \alpha \sum_{y \in \mathcal{S}^\delta} \Pr(y|x, u_{\min}(x))\hat{J}_{k-2}(y)$$
3. Run (for a prespecified length of time or number of steps, or until convergence) for each $x \in \mathcal{S}^\delta$ each D. P. r starting at $\hat{u}_r^*(x)$ from the previous iteration.
4. Send the new $\hat{u}_r^*(x)$'s to the upper D. P. level which recalculates
$$u_{\min}(x) = \arg\min_{\hat{u}_r^*(x)} \left\{ G(x, \hat{u}_r^*(x)) + \alpha \sum_{y \in \mathcal{S}^\delta} \Pr(y|x, \hat{u}_r^*(x))\hat{J}_{k-1}(y) \right\}$$
5. Set $u[k] = u_{\min}(x[k-1])$ and send to the plant.
6. Increase k and Go back to Step 1.

There are some points we wish to make about this algorithm. In practice, we suggest that the SLEK algorithm be run first, with random inputs, for a number of iterations in order for the network to begin converging. Then, the search for $u_{\min}(x)$ can be localized after a limited number of D. P. iterations. Thus, the number of required computations will decrease rapidly.

3 Computational Complexity

We will now discuss some computational complexity considerations. Let's assume we use N Gaussian RBFs in each dimension of the state-input space. Then each iteration of $\hat{u}_r^{(\cdot)}(x)$ at D. P. r takes $O(n^3 9^n)$ operations. At the upper level, each iteration of the SLEK algorithm takes $O(n^3 + 8^n)$ operations. Also, we need $O(N^m n^3 3^n)$ operations to calculate $u_{\min}(x)$.

4 Conclusions

To conclude, we have presented a hierarchical algorithm based on training recurrent networks for nonlinear control. The Extended Kalman Filter (EKF) algorithm and its localized variant (SLEK) play a central role, both in the training of the neural network and in the dynamic programming iterations to find the near optimal input at any time: The SLEK algorithm is used for training a neural network to approximate a nonlinear feedback system; we use a dynamic programming-based method of deriving near optimal control inputs for the real plant based on this approximation and a measure of its error (covariance). Finally, we combine these methods in a hierarchical algorithm for identification and control of a class of uncertain, unknown systems. We also provide computational complexities for the different calculations required for the algorithm.

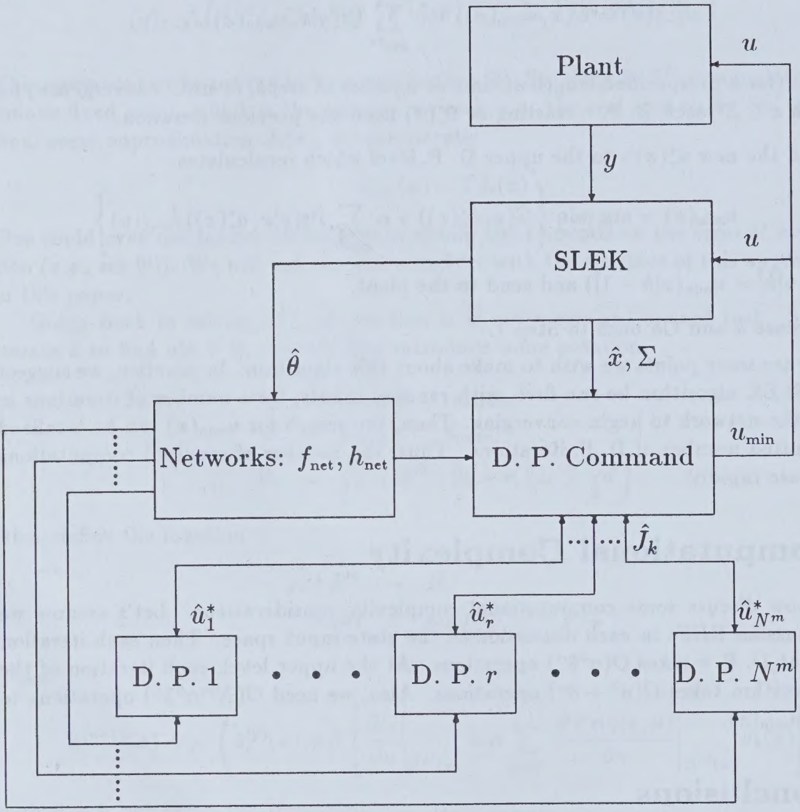

Plant
u
y
SLEK
u
$\hat{\theta}$
$\hat{x}, \Sigma$
Networks: $f_{\text{net}}, h_{\text{net}}$
D. P. Command
u_{min}
...... $\hat{J}_k$
$\hat{u}_1^*$
$\hat{u}_r^*$
$\hat{u}_{N^m}^*$
D. P. 1
D. P. r
D. P. N^m

Figure 1: Hierarchical Controller

5 Acknowledgements

During this work M. S. Branicky was supported by an Air Force Office of Scientific Research Graduate Fellowship (Contract F49620-86-C-0127/SBF861-0436, Subcontract S-789-000-056)., which is greatly appreciated.

References

[1] Brian D. O. Anderson and John B. Moore. *Optimal Filtering.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.

[2] C-S. Chow and J. N. Tsitsiklis. Optimal multigrid algorithm for continuous-state, discrete-time stochastic control. Technical Report LIDS-P-1751, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, February 1988.

[3] S. R. Chu, R. Shoureshi, and M. Tenorio. Neural networks for system identification. *IEEE Control Systems Magazine*, 10(3):31–35, 1990.

[4] M. M. Livstone, J. A. Farrell, and W. L. Baker. Computationally efficient algorithm for training recurrent networks. In *Proc. American Control Conference*, Chicago, IL, June 1992.

[5] M. B. Matthews and G. S. Moschytz. Neural network nonlinear adaptive filtering using the extended Kalman filter algorithm. In *Proc. Int. Conf. Neural Networks*, pages 115–119, Paris, France, July 1990.

[6] K. Narendra et al. Identification and control of dynamical systems using neural networks. *IEEE Trans. Neural Nets*, 1(1):4–26, March 1990.

[7] D. H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10(3):18–23, 1990.

[8] A. R. Odoni. On finding the maximal gain of Markov decision processes. *Operations Research*, 17:857–860, 1969.

The Logistic Lattice in Random Number Generation

Neal R. Wagner[1]
Division of Mathematics, Computer Science, and Statistics
University of Texas at San Antonio
San Antonio, TX 78249, USA

Abstract. This paper describes pseudo-random number generators based on non-linear dynamics. The recommended generator uses a special "re-mapped" form of the logistic equation at each of a finite set of nodes arranged in a 1- or 2-dimensional ring. Each node receives small perturbations from adjacent nodes by means of diffusive coupling equations—the so-called *coupled map lattice*. The generator is notable as the first to use non-linear dynamics in a lattice with a continuous state at each node. Time and space are discrete, but the state is continuous, unlike the cellular automata that Wolfram used to produce random numbers. It is a new source of pseudo-random numbers, unlike other current sources. The paper gives evidence that this generator possesses good statistical properties and a very long period, except in cases of negligible probability.

1. Introduction

Wolfram [11] used cellular automata and chaos theory to obtain pseudo-random numbers, but there has been little work on chaos methods utilizing floating point numbers, and such work was mostly carried out forty years ago.

An approach using iterative chaotic equations and floating point numbers must overcome several problems [2], [3], [8]:

(i) *Unknown distribution.* Usually the distribution of numbers produced by these non-linear equations is not known and probably not expressible in terms of known functions.

(ii) *Basins of influence of short cycles.* A number that gets close to a short cycle, even an unstable one, stays close for awhile (depending on how close it was initially), so the equations exhibit decidedly non-random behavior.

(iii) *No detailed quantitative theory.* Most of the theoretical results about these equations are qualitative and relate to global behavior.

(iv) *Theory only for real numbers.* The existing theory applies to infinite precision real numbers. Floating point numbers often behave quite differently.

This paper handles item (i) using a variation of the logistic equation, for which the distribution is known. Item (ii) requires that one iterate an equation long enough for the number to fill with noise before sampling a random number. Items (iii) and (iv) can only be approached with a statistical analysis of experimental data and with assumptions that floating point numbers will behave in some respects like infinite precision numbers.

---

[1] This material is based in part upon work supported by the Texas Advanced Research Program.

There has been a great deal of recent work on pseudo-random number generators, of which [1], [6], and [7] are typical. This paper describes a completely new generator, not currently studied by other researchers. The discussion focuses first on cycle length because that is the basic requirement of a reasonable random number generator. With the approach in this paper, the sequence is known to be equidistributed in infinite precision (see below). Assuming one iterates to fill the number with noise, one expects good statistical properties as long as the cycle length is long.

2. The Logistic Equation

The iterative equation $f(x) = 4x(1-x), 0 \leq x \leq 1$, known as the *logistic* equation, is historically interesting as one of the earliest proposed sources of pseudo-random numbers. Ulam and von Neumann suggested its use in 1947 [9], partly because it had a known algebraic distribution, so that iterated values could be transformed to the uniform distribution. The equation was mentioned again in 1949 by von Neumann [10] and much later in 1969 by Knuth [5, Exercise 3.4.1-24], but it was never used.

2.1. *Behavior in Infinite Precision*

Iterates of the "tent" function $g(u) = 1 - 2|u - (1/2)|, 0 \leq u \leq 1$ produce a uniform distribution, and g can be transformed to f via $T(u) = \sin^2((\pi/2)u)$. Both T and its inverse $T^{-1}(v) = (2/\pi)\arcsin(\sqrt{v})$ preserve the cycle structure of iterates of f and g.

If one is working with real numbers (infinite precision) then for all starting values except for a set of measure zero, iterates of g are equidistributed (= uniformly distributed) [5], [10]. Similarly, except for a set of measure zero, iterates of f are distributed according to $(1/\pi)\int_0^v dx/\sqrt{x(1-x)}$, and the sequence $\{T^{-1}(f_n(v))\}$ is equidistributed, again for any v outside a set of measure zero.

Despite being equidistributed, the basins of influence of the short cycles of g produce significant non-random behavior. A starting value very close to a value in a short cycle will stay close to that cycle for awhile, though all cycles are unstable, meaning that the values will eventually diverge. The cycle structure of f and g is complicated, with cycles of every finite length, and with countably many distinct starting values leading into each cycle (except for the cycle $(3/4)$). In fact the number of cycles of length n is equal to or a little less than $\mathbf{floor}((2^n - 1)/n)$, depending on the factorizations of $2^n \pm 1$.

2.2. *Behavior in Finite Precision*

In finite precision the behavior of f and g above is quite different from the theoretical results for infinite precision. Using a typical binary floating point unit, iterates of g from any starting value converge immediately to zero. (At each iteration, another significant bit becomes zero.) Iterates of f are more irregular, producing cycles as if successive values were chosen "at random." If successive values of f were chosen uniformly from n available numbers, then one would expect with probability 0.5 a duplicate value after about $1.18\sqrt{n}$ iterations and a cycle of length about $0.59\sqrt{n}$. In actuality the choices are not uniform, and the floating point numbers themselves are not uniformly distributed, but the behavior is not too far from cycles of the above lengths.

Table 1 shows the cycle lengths obtained by experiments using different kinds of hardware and precisions, as well as the percent of starting values leading into the cycle and the average initial run before the cycle starts. The cycle search in single precision is exhaustive

Hardware	Precision	Cycle Structure			Number of starting values
		Cycle length	Percent occurrence	Average initial run	
all types	single	**1**	**93.0%**	**2034**	4 194 305, all x such that $0.75 \le x \le 1$
		930	5.6%	340	
		431	1.0%	251	
		106	0.35%	244	
		205	0.1%	83	
		5	0.002%	31	
		4	0.0004%	7	
		3	0.00005%	2	
		1	0.00002%	0	
Weitek or Mips or Sparc	double	5 638 349	69.4%	54 000 000	7191
		1	**15.2%**	**10 000 000**	
		14 632 801	11.3%	8 500 000	
		10 210 156	1.5%	5 900 000	
		2 625 633	1.3%	3 800 000	
		2 441 806	1.2%	5 200 000	
		1 311 627	0.028%	240 000	
		960 057	0.014%	200 000	
VAX	double	86 058 517	84.0%	68 000 000	102
		1	**16.0%**	**44 000 000**	

Table 1: Cycles of the logistic equation (boldface = the cycle(0)).

because every cycle will eventually enter the range from $3/4$ to 1. Notable in these experiments are the small number of distinct cycles that appear, their short lengths, and the number of times the iteration falls into the cycle (0). (All cycle lengths are exact, but average initial runs have been rounded.)

The results in the table are hardware dependent. If the hardware differs in any way, or even in some cases if the computations are reordered or optimized, the results will be completely different in detail, but one still sees just a small number of short cycles. The entries for double precision do not use exhaustive search, so there are infrequently occurring cycles that are not listed. Each table entry refers to a unique cycle of the given length. It is possible to have two distinct cycles of the same length, but in this paper that only occurs with cycles of length 1: in Table 1, the cycle (0) (occurring 93% of the time), and the cycle $(3/4)$ (occurring 0.00002% of the time). The software checked that after getting a first cycle of a given length, each subsequent cycle of the same length had a number in common with the first one.

The frequent convergence to the cycle (0) can partly be explained by taking iterated inverse images under f from 0. In infinite precision one gets only countably many numbers leading into (0), so that the iteration falls into the cycle (0) with probability 0. Now switch to finite precision and consider the identity $f(0.5 + \epsilon) = 1 - 4\epsilon^2$. If ϵ is small, then $4\epsilon^2$ will be

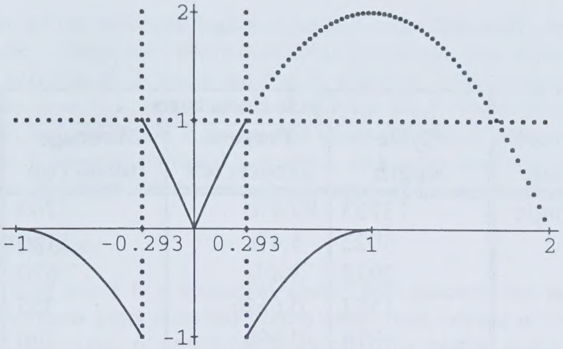


Figure 1: The original and re-mapped logistic equation.

very small, so that roundoff error will cause $1 - 4\epsilon^2$ to be exactly 1. Thus one gets a whole interval around 0.5 mapping to 1. Similarly there are two intervals mapping into the interval around 0.5, and so forth.

So far the logistic map sounds useless for pseudo-random number generation. For most starting values v, the sequence $\{T^{-1}(f_n(v))\}$ will be roughly equidistributed, but grossly non-random looking. Moreover, in finite precision there is a sizable probability of convergence to zero, and otherwise it may go into a fairly short cycle.

3. The Re-mapped Logistic Equation

The logistic equation yields numbers very close to 0 (on the positive side) and very close to 1. Available floating point numbers "pile up" near 0, but there is no similar behavior near 1. It is possible to restructure the equation so that values occurring near 1 are re-mapped to the negative side of 0. The following definition does this, mapping [-1, 1] to itself:

$$F_\beta = \begin{cases} 2|x|(2-|x|), & \text{for } |x| \le \beta, \\ -2(1-|x|)^2, & \text{for } \beta < |x| \le 1, \end{cases} \tag{1}$$

where $\beta = 1 - (1/\sqrt{2})$. A graph of the original (expanded by a factor of two) and re-mapped versions is shown in Figure 1.

In infinite precision, this re-mapped equation behaves exactly like the original, but with floating point numbers there is no longer any convergence to the cycle (0) of length 1. Unlike the original logistic equation, this version does not have an interval of numbers mapping into this cycle, so it is "well-tuned" to floating point numbers, since it requires and utilizes extra precision near 0 (on both sides).

Table 2 gives the cycles obtained by experimentation with the re-mapped equation. Notice that they are much longer on the average than the cycles of Table 1.

To transform numbers generated by $F_\beta(x)$ back to a uniform distribution one can use

$$S(x) = \begin{cases} (2/\pi)\arcsin(\sqrt{x/2}), & \text{for } 0 \le x < 1, \\ (2/\pi)\arcsin(\sqrt{-x/2}) + 0.5, & \text{for } -1 < x < 0. \end{cases} \tag{2}$$

Hardware	Precision	Cycle Structure			Number of starting values
		Cycle length	Percent occurrence	Average initial run	
all types	single	13753	89.9%	4745	8 388 609, all x such that $0.5 \leq x \leq 1$
		3023	5.4%	1150	
		2928	3.4%	670	
		1552	0.66%	355	
		814	0.6%	266	
		9	0.035%	191	
		1	0.00017%	14	
		3	0.000024%	1.5	
		1	0.000012%	1	
Weitek or Mips or Sparc	double	112 467 844	80.5%	105 000 000	1189
		61 607 666	5.7%	23 000 000	
		35 599 847	4.3%	19 000 000	
		1 983 078	3.6%	39 000 000	
		4 148 061	3.3%	60 000 000	
		15 023 672	2.5%	19 000 000	
		12 431 135	0.084%	5 500 000	
		705 986	0.084%	670 000	
M68040	double	73 573 097	80.6%	125 000 000	475
		38 326 216	16.8%	112 000 000	
		6 006 146	2.1%	34 000 000	
		5 195 797	0.42%	10 000 000	
Cray Y-MP	double	13 641 539	61.1%	27 500 000	3161
		1 281 377	20.5%	10 000 000	
		3 861 436	10.9%	7 000 000	
		1 630 338	2.9%	4 000 000	
		1 344 786	2.7%	2 800 000	
		6 034 850	1.1%	1 400 000	
		606 688	0.66%	2 000 000	
		782 542	0.095%	600 000	
		161 757	0.032%	200 000	
VAX	double	391 307 825	99.1%	284 000 000	110
		55 897 032	0.9%	45 000 000	

Table 2: Cycles of the re-mapped logistic equation.

An alternative to the remapped logistic equation is the "sawtooth" function: $h(x) = (mx)$ **mod** $1 = (mx) -$ **floor**(mx), where m is not an integer (the only interesting case). The author carried out experiments in single and double precision, with m equal to 1.5, with m taking on two values close to π, and with $m = 3\,141\,592\,653.7$. The resulting cycle lengths were very similar to those in Figure 2, except that round-off error produced short cycles for the large value of m. It seems likely that this equation, with some small non-integer m, would make an acceptable replacement for the remapped logistic equation in Section 4.

4. The Logistic Lattice

The *coupled map lattice* is a dynamical system with discrete time and space, and a continuous state. In recent years there has been a tremendous amount of research on these systems—hundreds of articles, of which [4] is typical. The models studied are usually in one or two dimensions, with "periodic boundary conditions," *i.e.*, the boundary wrapped in a circle or a torus. Researchers usually employ a logistic map at each lattice point and use *Laplacian* coupling. In this way they study a simple model for fluid mechanics which preserves the essential features of the partial differential equations, while remaining more tractable numerically. In one dimension the equations have the form:

$$x^i_{n+1} = f(x^i_n) + \nu[f(x^{i-1}_n) - 2f(x^i_n) + f(x^{i+1}_n)], \text{ for } 0 \le i < m,$$

Here m is the number of nodes, and ν is a constant known as the *viscosity*. All subscripts are calculated modulo m. The computation proceeds from step n to step $n+1$. It is important that ν be very small so that the equation will continue to have (almost) the same distribution as the unmodified logistic equation. Also small values for ν give the "fully developed turbulence" that is desirable for pseudo-random number generation. Combining terms gives:

$$x^i_{n+1} = (1-2\nu)f(x^i_n) + \nu[f(x^{i-1}_n) + f(x^{i+1}_n)], \text{ for } 0 \le i < m, \tag{3}$$

In two dimensions and after combining terms, the equation becomes:

$$\begin{aligned} x^{i,j}_{n+1} &= (1-4\nu)f(x^{i,j}_n) + \nu[f(x^{i,j-1}_n) + f(x^{i,j+1}_n) + f(x^{i-1,j}_n) + f(x^{i+1,j}_n)], \\ &\quad \text{for } 0 \le i < m, 0 \le j < m, \end{aligned} \tag{4}$$

The author has carried out various experiments with these equations, using for f the remapped logistic equation of section 3 (equation (1)), and using $\nu = 10^{-6}$ in single precision, $\nu = 10^{-14}$ in double precision, and larger values of ν for smaller precisions that are software simulated. In one dimension the experiments employed ring sizes m of 3, 4, and 5, and in two dimensions they used $m = 3$, *i.e.*, a 3×3 torus. The first basic property to study is the cycle length. This is found for a sequence $\{x_n\}$ by looking for the first value of n for which $x_n = x_{2n}$. (The cycle length is then a divisor of n.) Even in single precision a cycle never occurred, so one must be satisfied with a search for simpler events. Iterate either $\{x^i_n\}$ or $\{x^{i,j}_n\}$ for $n = 1, 2, 3, \ldots$ looking for one of the following events. (Here the *notation* is 1-dimensional, but the same definitions apply in the two-dimensional case.)

- *full hit* (or *m-hit*) event: $\{x^j_n\} = \{x^j_{2n}\}$, for all $0 \le j < m$, where n is fixed and as small as possible. (This is what one is looking for: a cycle whose length is a divisor of n.)
- *k-hit* event: $\{x^j_n\} = \{x^j_{2n}\}$, for k values of j, $k < m$. (This is only of interest in saying how likely a full hit might be.)

Configuration	Precision	# of iterations	# of hits	# of dups
1-dim, ring of 5	single	26.1×10^9	1531	3024
1-dim, ring of 5	double	39.7×10^9	0	0
2-dim, ring of 3 × 3	single	4.1×10^9	401	1667
2-dim, ring of 3 × 3	double	4.35×10^9	0	0

Table 3: Experiments with coupled equations.

- *single dup* event: $\{x_n^j\} = \{x_n^l\}$, for some $j \neq l$, and for some n.
- *dup* event: There are disjoint subsets $J_1, J_2, \ldots, J_k$ of $\{0, 1, \ldots, m-1\}$, where each J_i has size at least 2, and where for each i and for each $j_1 \neq j_2$ in J_i, it is true that $\{x_n^{j_1}\} = \{x_n^{j_2}\}$. (Dup events are significant because certain dups lead to a stable duplicated state.)
- *full dup* event: A dup event with $k = 1$ and $J_1 = \{0, 1, \ldots, m-1\}$.
- *k-dup* event: There are k duplicated pairs.
- *stable dup* event: a dup event where the duplicated nodes remain duplicated for all subsequent iterations. (Of course a full dup is stable. For $m = 3$, any single dup is stable. For $m = 4$, any single dup of opposite nodes is stable, while a single dup of adjacent nodes is not stable.)

Table 3 summarizes the results of one set of experiments, using $m = 5$ in one dimension, $m = 3$ in two dimensions, and the remappped logistic equation. In single precision, only single hit and single dup events occurred. In double precision there were no hit or dup events at all.

After finitely many iterations, there must be a cycle. This could occur as a single full-hit event without any initial stable dup event, but this outcome has negligible probability. Instead with high probability (essentially 1), there will be some stable dup event, *i.e.*, duplications continuing with all subsequent iterations.

THEOREM.

(a) *In a 1-dimensional ring of size m, the smallest initial stable dup event involves at least $k = \textbf{floor}((m-1)/2)$ duplicated pairs.*

(b) *In a 3×3 torus, the smallest initial stable dup event involves at least 3 duplicated pairs.*

PROOF (*Sketch of (a)*). Start with a ring of m nodes. Suppose there is an initial stable dup with a minimum number of duplications. This means there are two nodes with the same value repeated indefinitely. (For this argument, start with any duplicate pair.) Since these nodes must get the same perturbations at each iteration, they must each have two neighbors with the same values repeated indefinitely, and so on. (Actually, the *sum* of the values of perturbations must be the same, but this sum cannot stay the same unless the individual values are the same. Also a given perturbation could be zero once or a few times, but cannot remain zero indefinitely.) The initial duplicate pair splits the ring into two remaining segments of length i and j, both $\leq m-2$ and both ≥ 0. Suppose x below marks the original duplicated pair. Additional duplicated pairs on both sides of the x's can occur in two ways. They can be *nested*, like $(\ldots yxz \ldots zxy \ldots)$ or they can be *alternating*, like $(\ldots yxz \ldots yxz \ldots)$. In case m is odd, one of i or j must be even (includes 0) and the other odd. Here the simplest stable dup must be nested, with $(m-1)/2$ duplicated pairs, and one node left over on the odd side. In case m is even, there are three cases. One can have both i and j odd and $(m/2)-1$ nested pairs. Or one can have both i and j even (one possibly 0), with $m/2$ nested pairs. Finally, one can have $i = j$ and $m/2$ alternating pairs. This completes the proof.

In particular, the theorem says that a generator with at least $m = 7$ nodes in one dimension or 3×3 nodes in two dimensions requires three simultaneous duplicated pairs as the smallest initial stable dup event. In *single* precision, the results of Table 3 give estimates of very roughly 10^{-8} for the probability of a single hit at a given node or a single dup at a given pair of nodes. Because of the very small couplings involved, it is plausible to conjecture that actions at separate nodes are statistically independent of one another. Given this independence, one gets 10^{-25} as a very rough estimate for the probability of the first stable dup event, using single precision. Similar estimates for a full hit event before any stable dups give probability 10^{-56}, so one can discount the chance of this happening before a stable dup.

In double precision, the experiments in Table 3 produced no hit or dup events at all in 4.4×10^{10} iterations, so there are no estimates of their likelihood, but the probabilities are certainly very much smaller than in single precision. For practical purposes one can completely discount the chance of a stable dup or full hit event in double precision.

Extensive experiments were also run in software with simulated precisions of 8 and 12 significant bits, and were fully consistent with the above conclusions. Here as one would expect, the system eventually fell into an initial stable duplicated state. From such a state, further stable dups or a full hit were much more likely. Of course the system must terminate with a full hit and a cycle, but the full hit always involved only one or two separate values at all nodes.

5. Recommended Generator

The recommended generator, as coded in C in the Appendix, uses a coupled map lattice with at least $m = 7$ nodes in one dimension, or 3×3 nodes in two dimensions. It also uses the remapped logistic equation, and uses *decimation*, *i.e.*, values are sampled only after every 56 iterations, always from the same node. The "seed" for this random number generator is an array of m double-precision reals in the range from -1 to 1. Each separate array of m reals provides a different starting point for the generator. Some values are poor choices for a starting point: all the same value gives the equivalent of just a single logistic equation, and all 0's gives a fixed point. In practice, initializing the seed array using a auxiliary generator will work well, as shown in the Appendix. The author conjectures that the nodes are statistically independent of one another, so by sampling all nodes at once one would get a speedup by a factor of m.

As coded, the generator yielded good results when subjected to standard statistical tests. For example, the Kolmogoroff-Smirnov test was run with 10 000 sets of 1000 numbers each (for 10 000 000 iterations total), comparing the distribution of the 10 000 test results with the expected distribution, as described in [5]. Good performance on statistical tests is now an old-fashioned way to certify a pseudo-random number generator [6], but this is the best one can do with the generator described here.

6. Conclusions

This article has presented a new pseudo-random number generator based on an interesting problem from non-linear dynamics—namely the use of a variation of the logistic equation at each node of a coupled 1-dimensional or 2-dimensional lattice. There is statistical and theoretical evidence that this generator provides excellent pseudo-random numbers.

Acknowledgments

The author thanks Kay Robbins, Steve Robbins, Dave Eberly, Hugh Maynard, Marianne Cain, Paul Putter, and Stan Zietz for help. For at least four years of CPU time on various machines, thanks are due to Drexel University, the University of Texas at San Antonio, Sun Microsystems Inc., Digital Equipment Corporation, NeXT Computer, Inc., and the University of Texas System Center for High Performance Computing. Finally support came from Harold Longbotham and his Nonlinear Signal Processing Lab, and from the Texas Advanced Research Program.

References

[1] S.L. Anderson. "Random number generators on vector supercomputers and other advanced architectures." *SIAM Review 32* (2, June 1990) 221–251.

[2] P. Collet, and J.-P. Eckmann. *Iterated Maps on the Interval as Dynamical Systems.* Birkhaeuser, 1980.

[3] R.L. Devaney. *An Introduction to Chaotic Dynamical Systems, Second Edition.* Benjamin/Cummings, 1989.

[4] K. Kaneko. "Spatiotemporal chaos in one- and two-dimensional coupled map lattices." *Physica D 37* (1989) 60–82.

[5] D.E. Knuth. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms, Second Edition.* Addison-Wesley, 1981.

[6] S. Micali and C.P. Schnorr. "Efficient, perfect random number generators." *Advances in Cryptology, Lecture Notes in Computer Science 403.* Springer Verlag, 1990, 173–198.

[7] S.K. Park and K.W. Miller. "Random number generators: good ones are hard to find." *Comm. ACM 31* (1988) 1192–1201.

[8] C. Preston. *Iterates of Maps on an Interval.* Lecture Notes in Mathematics, No. 999, Springer Verlag, 1983.

[9] S.M. Ulam and J. von Neumann. "On combination of stochastic and deterministic processes" (abstract). *Bull. Amer. Math. Soc 53* (1947) 1120.

[10] J. von Neumann. "Various techniques used in connection with random digits." *Applied Math. Series 12,* National Bureau of Standards, 1951, 36–38.

[11] S. Wolfram. "Random sequence generation by cellular automata." *Advances in Applied Mathematics 7* (1986) 123–169.

Appendix. Source for the recommended generator (ANSI C).

The header file "random.h"

```
#define NMAX 7  /* NMAX should be >= 7 */
double chaotic_uniform (double [NMAX]);
/****************************************************************************
 * Generates uniformly distributed pseudorandom doubles in (0,1).           *
 * First initialize the array t. One could use an auxiliary generator ran() *
 *   that produces doubles in (0,1). Then invoke chaotic_uniform repeatedly.*
 * Thus the "seed" is an array t of NMAX doubles in (-1,1).  Like any       *
 *   seed for a RNG, the array t will change after each call.               *
 *   The applications program provides storage for the seed array t.        *
 * Typical usage by an application--to produce 100 random numbers:          *
 *      #include "random.h"                                                 *
 *      double ran(void);                                                   *
 *      double t[NMAX];                                                     *
 *      long i;                                                             *
 *      double result[100];                                                 *
 *      for (i = 0; i < NMAX; i++)                                          *
 *            t[i] = 2.0*ran() - 1.0;                                       *
 *      for (i = 0; i < 100; i++)                                           *
 *            result[i] = chaotic_uniform(t);                               *
 * The constant NMAX, the seed array t, and its initialization could all be *
 *   buried in the source file for unsophisticated users.                   *
 ***************************************************************************/
```

The source file "random.c"

```
#include <math.h>                        /* Needed for asin, sqrt and fabs */
#include "random.h"                      /* Header file                    */
#define BETA 0.292893218813452476        /* Magic number used in f         */
#define NU 1.0e-14                       /* Viscosity constant in step     */
#define TWO_DIV_PI 0.636619772367581343  /* 2/Pi used in S                 */
#define NSTEP 28                         /* Half the # of steps to iterate */
/*=========================================================================*/
static long mod (long i, long j)         /* If i is negative, then i%j     */
{                                        /* may be negative.  In general,  */
       long k = i%j;                     /* result is machine dependent.   */
       if (k < 0)  k = k + j;            /* Check your own architecture.   */
       return(k);
}
/*----------------------------------------------------Equation (1)---------*/
static double f (double x)                   /* Remapped logistic equation */
{
       double temp = fabs(x);
       if ( temp <= BETA ) return(2.0*temp*(2.0-temp));
       else return(-2.0*(1.0-temp)*(1.0-temp));
}
/*----------------------------------------------------Equation (3)---------*/
static void step (double t[], double tn[])          /* Coupled map lattice */
{
       long i;
       for (i = 0; i < NMAX; i++)  t[i] = f(t[i]);
       for (i = 0; i < NMAX; i++)
          tn[i] = (1.0-2.0*NU)*t[i] + NU*(t[mod(i-1,NMAX)]+t[mod(i+1,NMAX)]);
}
/*----------------------------------------------------Equation (2)---------*/
static double S (double x)               /* Change distribution to uniform */
{
       if (x >= 0) return(TWO_DIV_PI*asin(sqrt(x/2)));
       else return(TWO_DIV_PI*asin(sqrt(-x/2)) + 0.5);
}
/*-------------------------------------------------------------------------*/
double chaotic_uniform (double t[NMAX])      /* Iterate step 2*NSTEP times */
{
       long i;
       double tn[NMAX];                      /* Extra copy of seed array t */
       for (i = 0; i < NSTEP; i++) {
              step(t, tn);
              step(tn, t);
       }
       return (S(t[0]));
}
/*=========================================================================*/
```

A PARALLEL SCHEME
FOR THE CELL-TO-CELL BASED ANALYSIS OF DYNAMIC SYSTEMS

L. Fortuna(*), M. Lo Presti(**), G.Muscato (*), C. Vinci(*)

(*) Electric, Electronic and System. Depart., Viale A.Doria 6, 95125 Catania, Italy
(**) CoRi.M.Me., Stradale Primosole 50, 95121 Catania, Italy

Abstract.

The theory of Cell-to-Cell Mappings [1] has been receiving increasing attention in recent years as a new powerful method for global analysis of dynamic systems. With respect to the classical point to point methods, it better fits such problems deriving from numerical accuracy limitations and satisfies the necessity of analysing the dynamic behaviour in a more extended state space domain.

The suitability of the Cell-to-Cell algorithm is yet counterweighted by the great computational effort and the very time consuming procedure that characterizes it. Such considerations lead us to propose a parallel implementation of the algorithm.

In Section 1 a brief description of the Cell-to-Cell Mapping sequential algorithm is given; in Section 2 the proposed parallelization scheme is presented. Details on the adopted Transputer based architecture and some results about computational time improvements are reported in Section 3.

1 Cell-to-Cell sequential algorithm for the analysis of dynamic systems [2].

The main idea in the theory of Cell-to-Cell Mappings consists in dividing the state space domain of a dynamic system, limited by the range of interest of each variable, in a finite number of cells. A discretization of system dynamics is thus performed, since every point belonging to a cell is associated to the dynamic behaviour of the cell itself. For every cell then its global properties are investigated: starting form any point belonging to the cell, the analysis is carried out proceeding by successive integrations. The point $\mathbf{X}=(x_1, x_2,...,x_n)$ belongs to the cell $\mathbf{Z}=(Z_1, Z_2,...,Z_n)$ if x_i satisfies the following relation for all i:

$$\inf_i+(Z_i-0.5)h_i <= x_i < (Z_i+0.5)h_i+\inf_i$$

being h_i the cell size beside the x_i axis. The problem of how fine the scale of quantization of each variable axis must be has to deal with the accuracy of physical measurements and with numerical precision.

Suppose now the model be modeled by the dynamical equation:

$$\mathbf{X}(n+1)=\mathbf{F}(\mathbf{X}(n))$$

being $\mathbf{X}(n)$ the state variable vector, belonging to $\mathbb{R}^n$. Through the discretization of the state space domain the point mapping system is replaced by a Cell-to-Cell mapping $\mathbf{C}$, in accordance with the transformation:

$$Z_i(n+1)=C_i(\mathbf{Z}(n))=Int\{\frac{1}{h_i}(F_i(X(n)-inf_i)+\frac{1}{2}\}$$

where $\mathbf{Z}$ identifies the membership cell of $\mathbf{X}(n+1)$ and belongs to $\mathbb{N}$; Int denotes the larger integer which is less than or equal to the value in brackets; $\inf_i$ is the lower extreme of the range of x_i.

Let $\mathbf{C}^m$ be the mapping $\mathbf{C}$ applied m times to the cell $\mathbf{Z}$. A *periodic motion* P-K is constituted by a sequence of cells as

$$\mathbf{Z}(1)=\mathbf{C}^0(\mathbf{Z}(1)) \rightarrow \mathbf{Z}(2)=\mathbf{C}^1(\mathbf{Z}(1)) \rightarrow .... \rightarrow \mathbf{Z}(m+1)=\mathbf{C}^m(\mathbf{Z}(1)) \qquad (*)$$

satisfying the relation $Z(1)=\mathbf{C}^m(\mathbf{Z}(1))$(a loop). An *equilibrium cell* is therefore a P-1 motion.

A cell $\mathbf{Z}(j)$ is said to be *r-step removed* by a P-K motion if r is the minimum positive integer such that $C^m(\mathbf{Z}(j))=\mathbf{Z}(i)$ (a loop), being $\mathbf{Z}(i)$ any cell of the P-K motion.

As the domain of interest is divided into a finite number of cells, the analysis of the dynamic behaviour of every cell may lead to two possibilities: a cell may evolve towards or belong to a periodic motion constituted by cells of the domain of interest and is classified as a *regular cell*, or it may be mapped beyond the boundaries of interest in a region called *sink* and is classified as a *non regular cell*. All non regular cells correspond to an undesired behaviour of the system.

At the end of the investigation, the structural behaviour of every cell **Z** is summarized in three numbers: the *group number* Gr(**Z**), which distinguish the motion to which **Z** belongs; the *periodicity number* P(**Z**), indicating the number of cells constituting the loop of the motion to which **Z** belongs; the *step number* S(**Z**), the number of integration steps separating **Z** from the strictly periodic motion (if **Z** fully belongs to the periodic motion, S(**Z**)=0). If **Z** is a non regular cell, we assign conventionally Gr(**Z**)=1, P(**Z**)=1 and S(**Z**) equal to the number of integration steps separating **Z** from the sink.

During the algorithm of analysis, it is useful to distinguish three possible states for a cell. A cell **Z** which has not been yet called up is called *virgin cell*: its group number Gr(**Z**) is still zero. A cell which has been called up and is still under processing is said to be a cell *under processing* and has Gr(**Z**)=-1 as a symbol of identification. A cell whose global behaviour has been investigated is called *processed cell* and has positive group number.

The processing of a virgin cell involves a sequence as

$$\mathbf{Z} \rightarrow C(\mathbf{Z}) \rightarrow C^1(\mathbf{Z}) \rightarrow .... \rightarrow C^m(\mathbf{Z}) \qquad (*)$$

which concludes at the m-th mapping on a not virgin cell $\mathbf{Z}(m+1)$ that may be a sink cell, a processed cell or a cell under processing. In more details:

1) if $\mathbf{Z}(m+1)$ is a processed cell, whether a regular or a non regular one, then $Gr(C^j(\mathbf{Z}(1)))=Gr(C^m(\mathbf{Z}(1)))$, $P(C^j(\mathbf{Z}(1)))=P(C^m(\mathbf{Z}(1)))$ and $S(C^j(\mathbf{Z}(1)))=m-j+S(C^m(\mathbf{Z}(1))$ for $j=0,1,..,m-1$.

2) if $\mathbf{Z}(m+1)$ is a cell under processing, it must have had a previous occurrence in the sequence, let it be $\mathbf{Z}(i)$, then we assign a new group number, $P(C^j(\mathbf{Z}(1)))=m-i$, $S(C^j(\mathbf{Z}(1)))=i-j$ for $j=0,1,..,i-1$ and $S(C^j(\mathbf{Z}(1)))=0$ for $j=i,..,m-1$.

It may be observed that the sequential algorithm consists of a repetitive procedure of investigation over each cell. The generally high number of cells partitioning the domain is the cause of great computational efforts: analysis time increases with the order of the system and the number of subdivisions per axis.

2 Parallelization strategy.

The strategy of parallelization suggested in this work consists in partitioning the domain of interest in sub-domains and devoting sub-domain analysis to separate processors.

As a consequence of the partition, a sequence of cells starting in a sub-domain may be mapped into another subdomain. This case arises the problem of communication among processors in order to sew up the shared trajectory.

The idea of communicating as soon as in a processor a mapping into another subdomain occurs has been discarded: in fact, both processors have to be synchronized in order to communicate; moreover it may happen that the information required from the asking processor has not be still obtained in the analysis by the asked one.

The parallelized algorithm has been divided then into two parts: a first one of analysis in absence of communication and a second one of communication for sewing up the lacking informations. Let's indicate both the i-th process and the host processor by P_i. The two phases of analysis and sewing up constituting the parallel scheme are below described.

2.1 Phase of analysis.

Each processor P_i analyses the structural behaviour of every virgin cell in its sub-domain, building a sequence as (*). The investigation stops if $C^m(Z)$ belongs to one of the following cases:

1) it is a sink cell;
2) it is a cell under processing;
3) it is a processed cell of P_i;
4) it is a cell belonging to another subdomain (*pseudo-sink cell*);
5) it is a cell with non definitive informations.

In the first three cases the algorithm proceedes as described for the sequential one. In the fourth case temporary informations are assigned to all the cells in the sequence: in order to reconstruct the shared pattern during the subsequent phase of sewing up, every **Z**(i) must contain data about the coordinate and the domain of membership of the successive cell: we set Gr(Z(i))=-Z(i+1) and P(Z(i))=-k, where k indicates the domain of membership of **Z**(i+1) and the sign minus indicates that **Z**(i)'s data are not definitive ones. In the fifth case $C^m(Z)$ belongs to a pattern shared among more sub-domains. Evidently $C^m(Z)$ originates evidently from a previous case four. Cells belonging to cases four and five are denoted as *suspended cells*.

At the end of the phase of analysis, a procedure of sewing up starts the reconstruction of all non definitive processed cells.

2.2 Phase of sewing up.

It must be underlined now that at the end of the phase of analysis all cells belonging to periodic motions completely contained in only one sub-domain have definitely positive Gr and P numbers. Those cells that strictly belong or simply evolve towards to motions shared among more sub-domains have instead non definitive data. The strategy of sewing up is now described in the simple case of two processors. The reconstruction of the shared motions is carried out through the communication between the processors: the adopted strategy allows the processors to be always synchronized, as we foretold. All the cells in a subdomain have to be investigated to pick up the suspended ones. The process P_i reconstructs the suspended patterns basing on the informations contained in the group and periodicity data of every scanned cell having a negative Gr. If both the processors could ask informations to the other one with the same priority, in the case where a trajectory were mapped into another subdomain, a case of deadlock could occur. In order to avoid this occurrence, we assign to one processor only the possibility of being "asking processor", let it be P_1 for instance. During the phase of sewing up then, only P_1 analyses its own sub-domain, while P_2 waits for P_1's questions. When P_1 terminates, P_2's suspended remaining cells are only cells that evolves towards loops already processed during P_1's phase of analysis, (i.e. loops totally contained in P_1) or during P_1's phase of sewing up (shared loops).

From P_1's point of view, a sequence of cells under processing in the phase of sewing up may suspend on a $C^K(Z)$ cell such that:

1) $Gr(C^K(Z)) >= 1$;
2) $Gr(C^K(Z))$ = 'conventional negative number" (greater than the biggest coordinate value), indicating that $C^K(Z)$ is actually *under processing*;
3) $Gr(C^K(Z)) <= 0$ and $P(C^K(Z)) = -1$ or $P(C^K(Z)) = -2$, indicating that $C^K(Z)$ has been processed during the phase of analysis but has not yet definitive informations.

It is evident that more communications in both the directions are necessary to reconstruct the step and periodicity numbers of a shared trajectory, in the case of several mappings from P_1 to P_2 and viceversa. Thus at every mapping, informations have to be exchanged in both the directions.

For the reconstruction of the step number we proceedes as follows. Consider a shared motion where, for example **Z**(1)...**Z**(i) belong to P_1, **Z**(i+1)...**Z**(j) (j≥i+1) belong to P_2, **Z**(j+1)...**Z**(k) (k≥j+1) belong again to P_1. When **Z**(j) is mapped in P_1, P_2 temporarily stops its analysis and

send to P_1 the number of last processed cells, here j-i, so that P_1 could set S(**Z**(i) equal to this number. The same is done by P_1 when the sequence path is P_2-P_1-P_2. In this way the step number of all cells that are mapped into the opposite sub-domain is temporarily set.
When the sequence closes in a loop or evolves towards the sink cell, the step number is definitely reconstructed: for all cells belonging to the periodic motion S is set equal to zero; for the others the assignement is done as follows, starting from the cell that is 1-step removed from the sink or the periodic motion and going backwards:

$$S[\mathbf{Z}(i)]=S[\mathbf{Z}(i)]+S[\mathbf{Z}(i+1)]+1$$

The periodicity number of all cells under processing may be computed as the sum of loop's cells in P_1 and P_2 and is assigned to all the cells under processing.
The group number is assigned as "max group(P_1)+max group(P_2)-1" and the maximum group number of the processor where the loop has closed is increased by one.
During a communication phase a certain number of "synchronization variables" have to be exchanged to understand if the process P_i is sending or asking informations to P_j and how these informations have to be interpreted.

3 Results.

In order to validate the proposed approach, both the sequential and the parallel algorithm have been implemented over a T800 Transputer Architecture. This kind of microprocessors have been choosen for their suitability in parallel computation. We adopted a single T800 for the sequential version and two identical T800 for the parallel one. The analysis of a second order system governed by the equations:

$$x_1(n+1)=(1-\sigma)x_2(n)+(2-2\sigma+\sigma^2)[x_1(n)]^2$$
$$x_2(n+1)=-(1-\sigma)x_1(n) \qquad \sigma=0.1$$

has been considered. The ranges for x_1 and x_2 are [-1.01625, 2.525] and [-4.0, 3.03] respectively. In Fig. 1: computational times of the sequential (continuous line) and the parallel (dotted line) algorithm versus the total number of cells; in Fig. 2: the speed-up (T_{seq}/T_{parall}) versus the total number of cell subdividing the whole domain of interest (all times are in milliseconds).

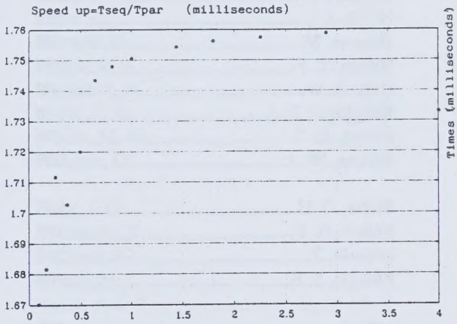


Fig. 1

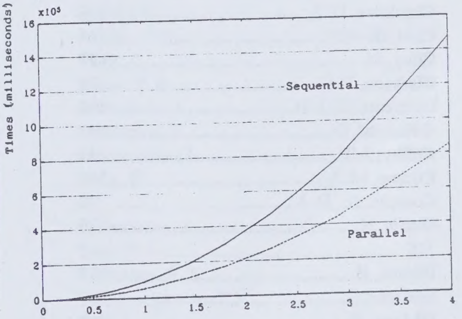


Fig. 2

Acknowledgement.
This work has been partially supported by Co.Ri.M.Me., Consortium for Microelectronics Researchs between the University of Catania and SGS Thomson (Italy).

References

[1] C.S.Hsu "A Theory of Cell-to-Cell Mapping Dynamical Systems", ASME Journal of Applied Mechanics, Vol.47, 1980, pp.931-939.
[2] C.S.Hsu, R.Guttalu " An Unravelling Alg. for Global Analysis of Dynamic Systems an Application of Cell-to-Cell Map." ASME Jour. of Appl. Mech. Vol.47, 1980, pp. 940-948.

INDEX OF AUTHORS

Aberer, K. 893
Agarwala, R. 594
Akl, S. G. 225
Andersland, M. S. 643
Andrews, M. G. 195
Argyropoulos, I. 465
Au, W.K.F. 826
Azemi, A. 349

Barnett, K. D. 167
Bean, N. G. 682
Bejczy, A. K. 702
Benhabib, B. 624
Bingulac, S. 83
Blake, I. F. 11
Blum, R. S. 289
Boel, R. K. 662
Bose, P. 402
Bradford, P. G. 185
Brandin, B. A. 624
Branicky, M. S. 913
Budge, S. E. 309
Burdick, J. 818

Casavant, T. L. 643
Caucal, D. 157
Chartrand, G. 543
Chen, C.-W. 98
Chen, S. L. 604
Chirikjian, G. S. 818
Choi, H.-A 424
Choy, M. 127
Clarkson, P. M. 71
Clements III, J. H. 495
Codenotti, B. 893
Coffey, J. T. 37
Corless, M. J. 267
Costello, Jr., D. J. 20
Cutaia, N. 279

Dathan, B. 137
DeCarlo, R. 330
De Iaco, B. 585
DeVoe, D. 742
Dehne, F. 373
Deller, Jr., J. R. 50
Deo, N. 215
Derin, H. 808
Doar, M. 682
Dokic, M. V. 71
Du, K. C. 515

Eom, K.-B. 505
Epley, D. L. 117

Fabien, B. C. 110
Fabri, A. 373
Fan, M.K.H. 248
Fernandez-Baca, D. 594
Fleisch, D. A. 60
Fortuna, L. 932
Francis, B. A. 692
Fürer, M. 205

Gannon, J. A. 643
Ghosh, B. K. 712
Goldberg, K. 836
Gonzalez, O. R. 112
Goodman, D. J. 446
Graham, P. 392
Grandhi, S. A. 446
Gray, J. E. 475
Greenberg, R. I. 664
Grigoriadis, K. M. 88, 359
Grunwald, G. 854
Gruska, D. P. 147

Hager, G. D. 854
Hanly, S. V. 426, 782
Hashmi, J. 543
Hegde, M. V. 652
Ho, T.-J. 650
Hossain, M. 404, 543
Huang, Y. F. 50
Hull, A. W. 60
Hunsinger, B. J. 60
Huynh, D. T. 157
Hwang, W.-J. 808

Ibarra, O. H. 575
Indeck, R. S. 792
Iwasaki, T. 267
Iyengar, S. S. 392

Jain, A. 215
Jankovic, M. 722
Juang, J.-N. 98

Kalpakis, K. 553
Kandala, S. 436
Khayrallah, A. S. 27
Khosla, P. 826
Knox, L. 742

Komo, J. J. 167
Krinsky, D. 174
Krishnamoorthy, M. S. 565
Kumar, S. P. R. 465

Lanterman, A. 329
Latombe, J.-C. 751
Lee, C. C. 174, 524
Lee, D. T. 195
Lee, S.-K. 424
Lei, M. 712
Leung, G. M. H. 692
Li, L. 277
Liu, D. 73, 485
Livstone, M. M. 913
Lo, J. T.-H. 495, 903
Longman, R. W. 98, 731
Lu, H.-I. 533
Luccio, F. 585
Lumpp, Jr., J. E. 643
Lund, M. A. 524

Mahalanobis, A. 39
Mahmoud, H. M. 874
Maier, R. S. 674
McCanna, J. 543
McDole, B. S. 475
McEliece, R. J. 455
Medidi, M. 215
Michel, A. N. 73, 369, 485
Miller, M. I. 279, 329, 515
Mills, D. G. 20
Min, P. S. 652
Misra, P. 241
Mitra, S. K. 39
Mixon, R. R. 864
Molnar, P. R. 15
Moon, T. K. 309
Muller, M. W. 792
Muscato, G. 932

Nagy, J. G. 257
Nakamura, T. 277
Nasipuri, A. 299
Nassar, M. 373
Nayar, S. K. 844

O'Sullivan, J. A. 279, 515, 792
O'Young, S. D. 633
Oh, H.-C. 664

Pagli, L. 585
Papadakis, T. 874
Paredis, C.J.J. 826
Park, J. 505

Pasupathy, S. 436
Peleties, P. 330
Petraglia, M. R. 39
Plemmons, R. J. 257
Porter, D. G. 792
Presti, M. L. 932

Raghavachari, B. 205
Rai, S. 884
Rajasekaran, S. 382
Ramamurthy, B. 565
Ramaswami, S. 382
Rao, A. 836
Rau-Chaplin, A. 373
Ravi, R. 533
Rohrs, C. E. 49
Roth, R. M. 1
Ryu, Y. S. 731

Sadowsky, J. S. 184
Saidi, H. 652
Sarnath, R. 235
Sawan, M. E. 340, 371
Schultz, R. R. 319
Shao, Z. 340, 371
Sherman, A. T. 553
Shermer, T. 402
Sherwani, N. A. 404, 543
Shin, Y. 117
Siegel, P. H. 1
Singh, A. K. 127
Sivarajan, K. N. 455
Skelton, R. E. 88, 115, 267, 359
Slutzki, G. 594
Snyder, D. L. 279, 329, 515
Song , S. 39
Sousa, E. S. 436
Srivastava, A. 279
Stevenson, R. L. 319
Stojmenovic, I. 225
Sung, W. 37

Tantaratana, S. 299
Tarn, T. J. 702
Teichman, R. S. 515
Theodosopoulos, T.V. 913
Tian, L. 157
Toussaint, G. 402
Tragoudas, S. 414
Trahan, J. L. 884
Tseng, D.-H. 98

Vaishnavi, V. K. 864
Valiveti, R. 373
VanLandingham, H. F. 83

Vannicola, V. C. 515
Vembu, S...799
Venkatesan, S..137
Verdú, S..799
Vigil, A. J...60
Vijayan, R. 446
Vinci, C. 932

Wagner, N. R. 922
Wang, D. X. 884
Wang, H. 575
Wang, K. 369
Whiting, P. 782
Wiegley, J. 836
Williams, K. J. 643
Wilson, R. H. 751
Wilson, S. G. 15
Wong, K. C. 614
Wonham, W. M. 604, 614, 624, 650

Xi, N...702

Yaghoobian, T. 11
Yaz, E...349, 359
Younce, R. C..49

Zhang, G. 742
Zheng, Q. 575
Zheng, S.-Q.................................392, 404
Zhu, B. 402
Zhu, G. 115

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1992
3 0112 008149285